BiLSTM-Guided Neuro-Symbolic Semantic Planning, Re-planning, and Backtracking for Efficient Emergency Evacuation

Ramzi Djemai

ORCID: 0009-0003-4636-3421

Director of Studies: **Assoc. Prof. Dr. Mohamed Chahine Ghanem**

Second Supervisors: Prof. Vassil Vassilev & Prof. Karim Ouazzane

A thesis submitted in fulfilment of the requirements of London Metropolitan University for the degree of Doctor of Philosophy in **Computing (Artificial Intelligence)**.

School of Computing and Digital Media
London Metropolitan University

April 2026

Declaration

I declare that the work presented in this thesis is my own, except where explicitly stated otherwise. This thesis has not been submitted for any other degree or professional qualification.

Signed: Ramzi Djemai

Date: 13/04/2026

Abstract

Indoor emergency evacuation under uncertainty requires planners that reason about building semantics, respond to dynamic hazards, and recover from routes that become infeasible during execution. Classical heuristic and dynamic search methods operate on purely metric representations and react to observed changes, but they do not exploit the semantic structure of the building and cannot anticipate when a chosen route is likely to fail. This thesis develops an ontology-guided neuro-symbolic framework that integrates a formal knowledge base with an incremental symbolic planner and a learned predictive component while preserving the guarantees of admissible heuristic search.

The symbolic layer is built on a knowledge base organised as topology, situations, actions, and events, expressed in OWL and refined through SWRL rules that derive admissible transitions and a semantic heuristic. A semantic variant of Lifelong Planning A* operates on the resulting state-transition graph and supports event-driven re-planning together with reactive backtracking when a dead-end is reached. A bidirectional LSTM with two heads is then introduced as a strictly subordinate learned layer. One head estimates residual cost and is combined with the semantic heuristic through a bounded blend that retains admissibility. The other head produces a calibrated backtracking probability that acts as a predictive gate on action selection without altering the set of admissible transitions.

The framework is evaluated on three structurally distinct building ontologies, a vertical tower, a multi-block hospital, and a multi-wing school, across a three-tier scenario taxonomy that varies hazard count and disruption severity. Seven planner configurations, including three blending weights of the proposed system and four established baselines, are compared on matched scenarios. The proposed system reduces mean evacuation time and node expansions relative to the strongest incremental baseline while maintaining success rate, and the observed improvements are consistent across the three buildings. The contribution is an integration pattern in which learned guidance accelerates symbolic search under uncertainty without eroding its formal properties, offering a reproducible basis for ontology-guided evacuation planning in dynamic indoor environments.

Keywords: ontology-guided evacuation planning; neuro-symbolic AI; BiLSTM

sequence learning; incremental heuristic search (LPA*); event-driven re-planning; predictive backtracking; indoor emergency evacuation.

Acknowledgements

I would like to express my sincere gratitude to my Director of Studies, Associate Professor Dr Mohamed Chahine Ghanem, without whom this thesis would likely have remained in a perpetual state of being nearly finished. His guidance, support, and continued encouragement were instrumental in ensuring that this work reached a final and complete form. I would also like to thank my second supervisor, Karim Ouazzane, for his support, mentorship, and encouragement from the very beginning of this journey. His guidance and support have made a real difference throughout.

I also want to acknowledge my previous Director of Studies, Professor Vassil Vassilev, for his early guidance and for the intellectual rigour he brought to this work. Our endless technical discussions and debates were both challenging and highly engaging, and I remain grateful for his persistence, insight, and genuine commitment to scientific enquiry.

A heartfelt thanks to Prof Preeti Patel for her support, mentorship, and belief in me throughout. Her coaching, encouragement, and the confidence she placed in me have had a lasting impact, and I am truly appreciative of everything she has contributed along the way.

To my family, in particular, my mother and my brother, your presence alone has meant and continues to mean more than I can ever put into words. Thank you for always being there for me.

Completing this research alongside a full-time career demanded sustained independence and self-direction, often across multiple competing priorities and prolonged periods of challenge and uncertainty. This thesis reflects both the research contribution and the persistence required to see it through.

To my partner, Olga, thank you for your constant belief and for the patience you have shown throughout this journey. It has been a long process, with inevitable interruptions along the way, and having you alongside me throughout has meant more than I can properly express. As I bring this important chapter to a close, even under partial observability and continual re-planning, the objective has remained unchanged. $\pi^* \longrightarrow$ YTL.

Publications

The following publications have been produced during the course of this research:

Journal Paper

- **Djemai, R.**; Kheddar, H.; Ghanem, M.C.; Ouazzane, K.; Nepomuceno, E. (2026). BiLSTM Guided LPA Planning, Re-Planning, and Backtracking for Effective and Efficient Emergency Evacuation. Smart Cities 2026, 9, 65. https://doi.org/10.3390/smartcities9040065. **(Q1, IF 5.5)**.

Conference Papers

- **Djemai, R.**, Vassilev, V., Ouazzane, K. and Dey, M., (2024). Knowledge-based reactive planning and re-planning–a case-study approach. In 2024 IEEE Conference on Artificial Intelligence (CAI) (pp. 770-775). IEEE. https://doi.org/10.1109/CAI59869.2024.00147.

- **Djemai, R.**, Vassilev, V., Ouazzane, K., Dey, M. (2025). Dynamic Path Planning for Indoor Evacuation Routing: A Universal Planning Framework. In: Bhateja, V., Dey, M., Simic, M. (eds) Intelligent Computing and Automation. FICTA 2024 2024. Smart Innovation, Systems and Technologies, vol 421. Springer, Singapore. https://doi.org/10.1007/978-981-96-0143-1_16.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
ABox	Assertional Box (OWL)
AUC	Area Under the Curve
BFS	Breadth-First Search
BiLSTM	Bidirectional Long Short-Term Memory
BIM	Building Information Modelling
DL	Description Logic
IFC	Industry Foundation Classes
LPA*	Lifelong Planning A*
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
OWL	Web Ontology Language
RDF	Resource Description Framework
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
SWRL	Semantic Web Rule Language
TBox	Terminological Box (OWL)

1 Introduction

Indoor emergency evacuation is a safety-critical domain in which planning decisions must be made rapidly, under uncertainty, and in environments where conditions change dynamically as hazards evolve. When a fire breaks out on the fifth floor of a ten-storey building, the occupants must navigate a complex spatial structure, corridors, stairwells, lobbies, and exits, while smoke propagation, structural failures, and congestion alter the viability of previously safe routes. The planner guiding such an evacuation must integrate knowledge about the building's topology, the semantics of evacuation actions, the classification of dynamic hazard events, and the anticipated costs of traversing the evolving environment. No single computational paradigm, whether purely symbolic reasoning, classical heuristic search, or data-driven machine learning, addresses all of these requirements simultaneously.

This thesis presents a neuro-symbolic framework that integrates ontology-based domain modelling, incremental heuristic search, and neural cost estimation with predictive backtracking to address the challenge of indoor emergency evacuation planning under uncertainty. The framework is designed so that symbolic and learned components operate within clearly defined roles: ontological models provide domain grounding, formal guarantees, and interpretability; neural networks provide trajectory-aware cost refinement and anticipatory decision-making; and incremental search provides efficient re-planning when environmental conditions change. The bounded integration of these components ensures that the neural layer cannot compromise the optimality guarantees of the underlying search algorithm.

1.1 Problem Statement and Motivation

Emergency evacuation in buildings remains a significant public safety concern. Building fires alone account for thousands of fatalities annually worldwide, and the majority of fire-related deaths occur in indoor environments where occupants are unable to reach exits before conditions become untenable [81]. The problem is compounded by the structural complexity of modern buildings, multi-storey layouts, restricted access zones, and limited vertical circulation, and by the dynamic nature of emergency events that alter the traversability of the building

during evacuation.

Three fundamental challenges characterise indoor emergency evacuation planning.

Challenge 1: Semantic complexity of indoor environments. Indoor spaces are not simply geometric containers; they carry semantic meaning that constrains evacuation behaviour. A staircase connects floors but imposes capacity limits; a fire door may be passable in one direction but not the other; a hospital ward contains patients with restricted mobility who cannot use standard evacuation routes. Capturing these semantics requires a representation that goes beyond the geometric graph models traditionally used in path planning. Ontological knowledge representation, specifically OWL class hierarchies combined with SWRL inference rules, provides a formal, machine-readable encoding of such semantics, enabling the planner to reason about what actions are permissible in which situations rather than simply computing shortest geometric paths.

Challenge 2: Dynamic re-planning under evolving hazards. Emergency events are inherently dynamic: fires spread, corridors become obstructed, exits are blocked by debris or smoke. A path that is optimal at the moment of alarm may become infeasible minutes later. Classical search algorithms such as A* [34] compute a complete plan from scratch and must discard all computational work when conditions change. Incremental heuristic search algorithms such as LPA* [52] address this limitation by propagating cost changes through locally inconsistent nodes, repairing only the affected portion of the search tree rather than re-computing the entire solution. This efficiency is critical in evacuation scenarios where multiple hazard events may alter edge costs during a single evacuation episode.

Challenge 3: Anticipatory cost estimation and predictive backtracking. Even with incremental re-planning, the planner's performance depends on the quality of its heuristic function. A purely symbolic heuristic, such as the semantic heuristic $h_o(n)$ that counts minimum SWRL rule chain transitions to the goal, provides a valid lower bound but may not capture the trajectory-specific cost patterns that distinguish easy evacuations from difficult ones. Furthermore, standard incremental search algorithms react to dead-ends only after reaching them: the planner discovers that a path is blocked only when it arrives at the blocked node. In confined indoor environments where backtracking through narrow

corridors incurs significant time penalties, anticipating the need for backtracking *before* a dead-end is reached can substantially reduce evacuation time. Neural sequence models, trained on evacuation trajectories, offer a data-driven approach to both cost refinement and predictive backtracking.

Research problem. The problem addressed by this thesis is how to compute indoor emergency evacuation plans that remain semantically valid and computationally efficient as hazard events alter transition feasibility and traversal costs during execution, while exploiting predictive information to reduce avoidable backtracking without compromising the admissibility of the underlying heuristic search or the constraints encoded in the domain model.

These three challenges motivate the neuro-symbolic approach presented in this thesis: an integration of ontological domain models (addressing Challenge 1), incremental heuristic search with event-driven re-planning (addressing Challenge 2), and BiLSTM-based neural cost estimation with predictive backtracking (addressing Challenge 3).

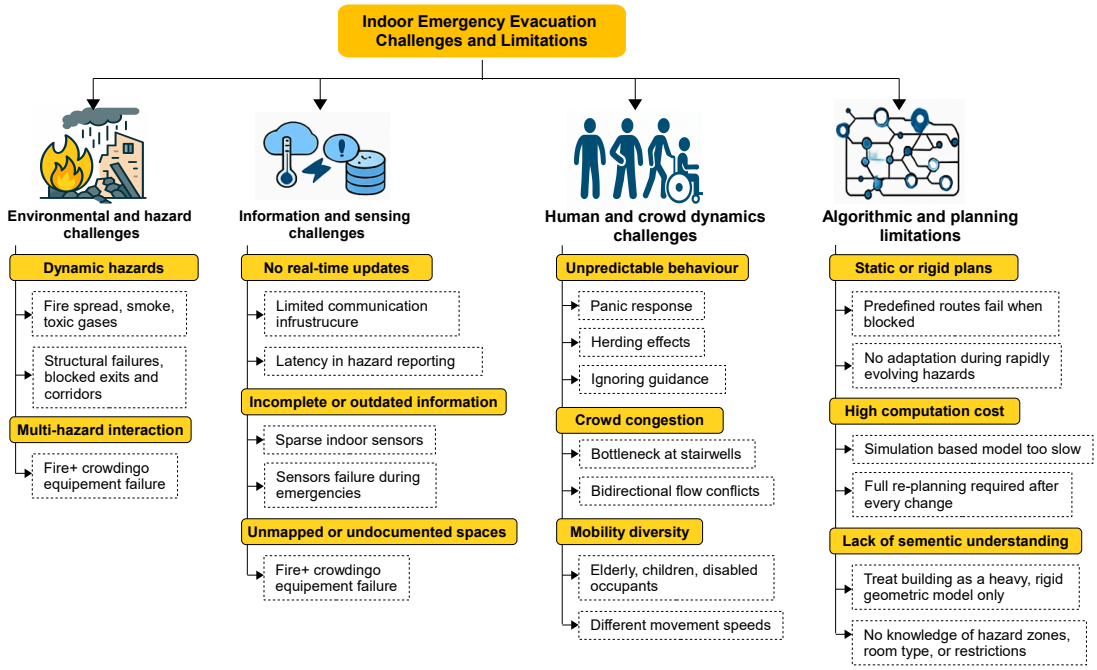

Figure 1.1: Key challenges in indoor emergency evacuation, including dynamic hazards, incomplete or unreliable sensing, human and crowd behavioural uncertainty, and limitations of existing planning algorithms.

1.2 Research Context

The research presented in this thesis sits at the intersection of three active fields: evacuation planning, knowledge representation and reasoning, and neuro-symbolic artificial intelligence.

Evacuation planning. The evacuation planning literature spans a spectrum from microscopic pedestrian simulation models [69] through agent-based approaches [81] to optimisation-based methods [27, 28]. More recent work has explored machine learning approaches, including reinforcement learning for route guidance [79, 90] and deep learning for pedestrian behaviour prediction [11, 94]. However, the majority of these approaches operate on geometric or grid-based spatial representations that do not encode the semantic constraints governing evacuation behaviour. Ontology-based approaches [20, 70] address this gap but have typically been limited to static knowledge representation without integration into dynamic planning algorithms.

Knowledge representation for indoor spaces. The representation of indoor environments has progressed from simple grid models through graph-based topological models to semantic models based on ontological formalisms. Standards such as IndoorGML [49] provide structured representations of indoor topology, and Building Information Modelling (BIM) offers rich architectural data. OWL and SWRL enable the encoding of not only spatial connectivity but also domain-specific rules governing actions, constraints, and situation classifications. The knowledge base $K = (\mathcal{T}, \mathcal{S}, \mathcal{A}, \mathcal{E})$ employed in this thesis, comprising Topology, Situations, Actions, and Events ontologies, extends prior ontological models by providing a complete operational domain model over which planning algorithms can directly operate.

Neuro-symbolic AI. The integration of symbolic reasoning and neural learning has emerged as a major research direction in artificial intelligence [25, 39]. Neuro-symbolic systems seek to combine the strengths of symbolic AI (interpretability, formal guarantees, domain knowledge encoding) with those of neural networks (pattern recognition, generalisation from data, handling of uncertainty). In the context of planning, neuro-symbolic approaches use learned models to augment symbolic planners, for example, by providing learned heuristics that improve

search efficiency while the symbolic layer maintains correctness guarantees. The framework presented in this thesis instantiates this principle: the BiLSTM provides learned cost estimates and backtracking predictions that improve search efficiency, while the ontological model and the bounded heuristic blend ensure that admissibility is preserved.

Research programme. This thesis is the culmination of a research programme that has progressed through three published stages. The foundational stage [22] established four interlinked ontological models and demonstrated LPA* planning with a semantic heuristic over a Tower Building case study. The intermediate stage [21] extended this foundation with event-driven reactive re-planning, a three-tier scenario taxonomy (Simple, Semi-Complex, Complex), and case study evaluation. The current stage [23] completes the neuro-symbolic framework by adding BiLSTM-based neural cost estimation, predictive backtracking, and a bounded hybrid heuristic that integrates learned and semantic components. Each stage addresses a distinct limitation of the preceding one, and the thesis presents the complete integrated framework.

1.3 Research Questions

The research is guided by four questions that correspond to the principal components of the framework.

RQ1: How can building topology and evacuation semantics be formally encoded in a machine-readable knowledge representation that supports both static planning and dynamic re-planning?

RQ2: Can an incremental heuristic search algorithm, guided by ontology-derived semantic information, provide efficient evacuation planning in dynamic indoor environments?

RQ3: Can a learned neural component augment the symbolic planner to improve evacuation efficiency without compromising the formal guarantees of the underlying search algorithm?

RQ4: Does the proposed neuro-symbolic framework generalise across structurally distinct building topologies?

RQ1 addresses the knowledge representation challenge and is answered by the ontological domain models presented in Chapter 3. RQ2 addresses the incremental search and re-planning challenge and is answered by the semantic heuristic and event-driven LPA* adaptation presented in Chapter 4. RQ3 addresses the neural augmentation challenge and is answered by the BiLSTM dual-head architecture and bounded hybrid heuristic presented in Chapter 5. RQ4 addresses the generalisability challenge and is answered by the cross-building instantiation and evaluation design presented in Chapters 3 and 6.

1.4 Research Methodology Overview

The research follows a design science methodology in which the primary artifact is the neuro-symbolic evacuation planning framework. The methodology comprises four phases.

Phase 1: Domain modelling. The indoor evacuation domain is formalised through four interlinked OWL/SWRL ontologies that encode building topology, occupant situations, navigation actions, and hazard events as a machine-readable knowledge base $K = (\mathcal{T}, \mathcal{S}, \mathcal{A}, \mathcal{E})$. The ontological models are designed with a TBox/ABox separation that enables reuse: the generic TBox (including 47 SWRL rules) defines the domain vocabulary and transition rules, while building-specific ABox instances populate the topology for each case study building.

Phase 2: Algorithm design. The LPA* incremental heuristic search algorithm is adapted to operate over the state-transition graph derived from the ontological model. A novel semantic heuristic $h_o(n)$ is designed that counts the minimum SWRL rule chain transitions from the current situation to the goal, and its admissibility is formally proven. An event-driven re-planning architecture is designed that classifies dynamic hazard events using the ontology and triggers appropriate planner responses, from incremental edge-cost updates to full backtracking.

Phase 3: Neural augmentation. A BiLSTM with a dual-head architecture is designed and trained on synthetic evacuation trajectories generated from the ontological simulator. The cost estimation head provides remaining cost predictions that refine the semantic heuristic into a hybrid heuristic $h'(n) = \min\{h_o(n),\ (1 - \lambda)\, h_o(n) + \lambda\, h_l(n)\}$, and its admissibility is formally proven. The backtracking

prediction head provides calibrated probabilities that trigger predictive path reversal when the predicted backtracking probability exceeds a threshold τ^*.

Phase 4: Evaluation. The framework is evaluated through a comparative study of seven algorithm configurations across three building topologies. The evaluation employs a three-tier scenario taxonomy (Simple, Semi-Complex, Complex) with 50 scenarios per building, and measures evacuation time (T_{avg}), success rate (S_r), expanded nodes (N_{nodes}), and BiLSTM prediction performance (RMSE, AUC-ROC, accuracy). Six research hypotheses are formulated and tested through the Tower Building evaluation, with Hospital and School evaluations designed to assess cross-building generalisation.

1.5 Contributions

The thesis makes five principal contributions to the field of indoor emergency evacuation planning.

The core contribution of this thesis is the integration of established symbolic and learned techniques within an ontology-guided neuro-symbolic framework for dynamic indoor emergency evacuation planning. OWL, SWRL, LPA* and BiLSTM are not claimed as novel techniques in isolation. Rather, the contribution lies in their domain-specific adaptation and bounded integration: the ontology and rule layer define and constrain the permissible state-transition structure, the incremental search layer supports efficient re-planning when transition feasibility or traversal costs change, and the learned component supplies predictive guidance without overriding ontology-defined transition constraints or compromising the admissibility properties of the underlying heuristic search. The five contributions listed below constitute the principal elements through which this integrated contribution is realised.

1. **Ontological domain models for indoor evacuation.** A suite of four interlinked OWL/SWRL ontologies, Topology, Situations, Actions, and Events, that encode building structure and evacuation semantics as a machine-readable knowledge base $K = (\mathcal{T}, \mathcal{S}, \mathcal{A}, \mathcal{E})$. The architectural transferability of the framework is demonstrated through instantiation across three structurally distinct buildings, a 10-floor Tower, a 3-storey Hospital, and a 2-storey School, each reusing the same 47-rule generic SWRL core with

building-specific extension layers.

2. **Semantic heuristic and incremental search.** The adaptation of LPA* to operate over the ontology-derived state-transition graph, together with a novel semantic heuristic $h_o(n)$ that counts the minimum SWRL rule chain transitions from the current situation to the goal. The admissibility of $h_o(n)$ is formally proven, ensuring optimal paths under the current edge weights. The event-driven re-planning architecture classifies dynamic hazard events using the ontology and triggers appropriate planner responses.

3. **BiLSTM dual-head neural augmentation.** A Bidirectional LSTM with a dual-head architecture providing remaining cost estimation (refining the semantic heuristic into a hybrid heuristic $h'(n)$) and backtracking probability prediction (enabling predictive path reversal before dead-ends are reached). The bounded hybrid heuristic is proven to preserve admissibility for all values of $\lambda \in [0, 1]$. Admissibility here is a search-theoretic property rather than a guarantee of real-world occupant safety; this distinction and its implications are discussed explicitly in Section 6.9.2.

4. **Neuro-symbolic integration architecture.** A principled integration pattern in which the symbolic layer (ontology, SWRL rules, semantic heuristic) operates independently from the learned layer (BiLSTM). The learned component influences only node expansion ordering through the bounded heuristic blend and backtracking decisions through the calibrated threshold τ^*. The BiLSTM does not modify the admissible transition set, create new actions, or bypass safety constraints encoded in the SWRL rules.

5. **Comparative evaluation framework.** A systematic evaluation comparing seven algorithm configurations (three BiLSTM-guided LPA* variants, Conventional LPA*, A*, D*, and D* Lite) across three building topologies with a three-tier scenario taxonomy, six research hypotheses, and ablation analysis demonstrating the independent contributions of the cost estimation and backtracking prediction heads.

Table 1.1 maps these contributions to the thesis chapters in which they are developed.

Table 1.1: Mapping of contributions to thesis chapters.

Contribution	Description	Chapter(s)
C1	Ontological domain models	Chapter 3
C2	Semantic heuristic and incremental search	Chapter 4
C3	BiLSTM dual-head neural augmentation	Chapter 5
C4	Neuro-symbolic integration architecture	Chapters 4 and 5
C5	Comparative evaluation framework	Chapter 6

1.6 Thesis Structure

The remainder of this thesis is organised as follows.

Chapter 2: Literature Review. Provides a critical review of the literature across five areas: indoor evacuation planning approaches, knowledge representation for indoor environments, heuristic search algorithms for dynamic environments, machine learning for evacuation, and neuro-symbolic AI. Identifies the research gaps that motivate the proposed framework.

Chapter 3: Ontological Domain Modelling for Indoor Evacuation. Presents the four-ontology knowledge base $K = (\mathcal{T}, \mathcal{S}, \mathcal{A}, \mathcal{E})$, including the Topology, Situations, Actions, and Events ontologies, their OWL class hierarchies, SWRL rule categories, and the state-transition graph formalism. Demonstrates the instantiation across three building topologies.

Chapter 4: Semantic Heuristic Search and Event-Driven Re-planning. Presents the adaptation of LPA* to the ontology-derived state-transition graph, the semantic heuristic $h_o(n)$ with its admissibility proof, and the event-driven re-planning architecture that classifies hazard events and triggers appropriate planner responses.

Chapter 5: BiLSTM-Guided Planning with Predictive Backtracking. Presents the BiLSTM dual-head architecture, the training data pipeline, the bounded hybrid heuristic $h'(n)$ with its admissibility proof, and the predictive backtracking mechanism with threshold calibration.

Chapter 6: Experimental Evaluation and Discussion. Presents the experimental setup, evaluation metrics, six research hypotheses, and results from the comparative evaluation of seven algorithm configurations on the Tower Building. Includes ablation analysis, BiLSTM prediction performance assessment, and discussion of findings and limitations.

Chapter 7: Conclusion and Future Work. Summarises the contributions, revisits the research questions in light of the experimental evidence, identifies directions for future work including multi-occupant coordination, real-time sensor integration, and alternative neural architectures, and offers concluding remarks.

Appendices. Three appendices provide supplementary material: Appendix A contains the core algorithm pseudocode; Appendix B contains SWRL rule examples; and Appendix C contains representation examples. Evaluation setup tables (building topology, edge cost model, algorithm configurations, scenario taxonomy, and training–test split) are consolidated in Chapter 6 rather than in a separate appendix.

2 Literature Review

2.1 Introduction

Emergency evacuation planning from complex indoor environments is an old problem with a deceptively modern character. Classical approaches,Dijkstra-based shortest path algorithms, static network models, aggregate flow models,have proven adequate for small-to-medium buildings and for offline planning scenarios. However, three converging demands have rendered these approaches insufficient for contemporary practice.

First, the *semantic complexity* of modern buildings has grown dramatically. Contemporary buildings are heterogeneous ecosystems: office towers mix open-plan spaces with restricted-access labs; hospitals intermingle general wards with ICUs where evacuation follows building-specific protocols; schools and transit hubs must enforce distinct routing rules for children, staff, and the mobility-impaired. A planning algorithm that treats all nodes uniformly, assigning a single floating-point cost to each transition, discards the semantic information required to compute a feasible, safe, and policy-compliant route.

Second, the *dynamic environment* assumption is now foundational, not optional. Modern emergency response systems integrate real-time sensor networks, occupancy prediction, and adaptive resource allocation. The planner must respond to changing conditions,fires, blocked exits, power failures, congestion,not by recomputing a new plan from scratch but by *re-planning incrementally*, preserving prior computation where applicable and focusing search effort only on affected regions of the plan.

Third, *learned cost estimation* has emerged as a critical capability. The traversal cost of a spatial transition depends not only on geometric distance but on human behaviour under stress, congestion dynamics, accessibility constraints, and environmental factors. No hand-crafted cost model can capture this complexity; learned models trained on evacuation data offer a principled alternative. Yet learned models themselves are opaque: they make no commitment to admissibility, feasibility, or policy compliance. This creates a fundamental tension at the heart

of modern evacuation planning: how to leverage learned cost estimates without sacrificing the formal guarantees that classical planning provides.

This chapter surveys the literature at the intersection of these three demands. It is organised in six parts:

1. **Section 2.2** reviews classical and modern approaches to indoor evacuation planning, identifying how existing work treats routing, simulation, and multi-agent coordination.
2. **Section 2.3** surveys knowledge representation methods for built environments, with emphasis on ontologies and formal spatial reasoning.
3. **Section 2.4** examines heuristic search algorithms for dynamic environments, focusing on incremental approaches.
4. **Section 2.5** surveys machine learning methods for navigation and cost estimation, with emphasis on sequence models.
5. **Section 2.6** reviews neuro-symbolic AI approaches and their application to planning.
6. **Section 2.7** performs a critical gap analysis, demonstrating that no existing work combines all four required capabilities: ontology-based domain encoding, incremental search, learned cost estimation, and predictive re-planning.

The chapter concludes by positioning the contributions of this thesis (Chapter 3, Chapter 4, Chapter 5, Chapter 6) at this intersection.

2.2 Indoor Emergency Evacuation Planning

2.2.1 Classical Approaches to Evacuation Routing

Classical evacuation planning algorithms are rooted in the graph-based shortest-path paradigm. The building is modelled as a weighted directed graph where nodes represent spaces (rooms, corridors, exits) and edges represent adjacencies. Edge weights are typically geometric distances, sometimes augmented by simple congestion penalties. Planning is then reduced to path-finding: compute the

shortest path from current location to a safe exit, and return this path to the evacuee.

Seminal work in this tradition includes the agent-based models of Shi et al. [81] and Nishinari et al. [69], which combine microscopic pedestrian dynamics with simple graph-based routing. These models treat the building as a 2D cellular automaton or Voronoi diagram and apply gradient-descent-like heuristics to guide agents towards exits, and they trace their theoretical lineage to the foundational panic-dynamics model of Helbing et al. [38] and the heuristic-search framework developed by Pearl [73] and Pohl [75]. The strength of these approaches lies in their computational efficiency: with small node counts and uniform costs, Dijkstra's algorithm runs in near-linear time, a property exploited by Cimellaro et al. [18] for large-scale high-rise routing.

However, classical approaches have four fundamental limitations.

First, they ignore semantic diversity. A shortest path that is geometrically valid may violate fire codes, accessibility regulations, or building-specific evacuation protocols. For example, a hospital wing may restrict ICU patients to specific exits; a school may mandate that younger children use primary exits while older students use secondary exits. A routing algorithm that assigns each exit the same intrinsic desirability cannot encode these constraints without explicit pre-processing or path post-filtering.

Second, they assume static environments. Classical algorithms compute a plan once at the start of evacuation. In reality, conditions change: a fire spreads, an exit becomes impassable, occupancy data arrives, or sensors detect a structural hazard. The planner must re-plan, but classical shortest-path approaches offer no mechanism to reuse prior computation. They either recompute from scratch (wasteful) or cache the old plan and assume it remains valid (unsafe).

Third, they conflate routing and coordination. Early work by Goerigk et al. [28] and Goerigk and Grün [27] formulated evacuation as a min-cost flow problem, computing optimal routing for aggregate flows. These models improve upon single-path algorithms but treat all occupants symmetrically and assume perfect information and perfect compliance. They do not account for individual heterogeneity, information asymmetry, or the stochastic nature of human behaviour under stress.

Fourth, cost models are hand-crafted and opaque. The traversal cost assigned to each edge is typically a function of distance, width, and perhaps a

fixed congestion multiplier. This cost model is specified a priori and never updated. It makes no attempt to learn from historical evacuation data, simulations, or real-world incidents. As a result, cost estimates may be wildly inaccurate in practice, leading to suboptimal or infeasible routes.

Modern classical algorithms (e.g., those surveyed by Gul et al. [32]) address some of these limitations through hybrid approaches: incorporating time-dependent edge weights, allowing multiple simultaneous paths, or coupling shortest-path algorithms with agent-based simulations. However, they do so ad hoc, lacking a unifying framework that integrates semantic constraints, dynamic re-planning, and learned costs.

The limitations above expose a structural trade-off between graph-based planning and simulation-based approaches that pervades the evacuation literature. Graph-based planners (Section 2.4.1) operate on an explicit state space and can, in principle, provide optimality guarantees: A^* returns a least-cost path if the heuristic is admissible and the edge costs are correct. However, they require both conditions to hold simultaneously,admissibility of $h(n)$ and correctness of $c(n, n')$,and classical evacuation models satisfy neither robustly. Edge costs are hand-crafted from geometric distance and fixed congestion multipliers, so they diverge from true traversal cost as soon as occupant density changes (a congestion event multiplies the true edge cost) or a hazard invalidates an edge (fire removes a corridor from the feasible graph). Without a mechanism to update $c(n, n')$ in real time, the optimality guarantee is formal but vacuous. Simulation-based methods (Section 2.2.2), conversely, can model dynamic edge-cost changes and node invalidations with high fidelity,each agent senses congestion, fire, and structural damage at every time step,but they operate by forward rollout of local agent rules, not by backward induction over a cost function. They therefore provide no global cost accounting, no admissibility property, and no guarantee that the emergent behaviour constitutes an optimal or even feasible evacuation plan. The two paradigms are complementary in their strengths and weaknesses, yet no work in the classical evacuation literature provides a principled interface between them: the graph planner cannot consume the simulator's dynamic state, and the simulator cannot exploit the planner's optimality machinery.

2.2.2 Simulation-Based Methods

Recognising the limits of pure shortest-path routing, the evacuation literature has adopted simulation as a complementary approach. Simulation-based methods model the building, the occupants, and the environment at high fidelity, execute a synthetic evacuation, and measure aggregate outcomes such as total evacuation time, maximum occupancy, or fatality counts.

The EVACNET framework of Goodwin et al. [29] applies ant-colony optimisation to route planning within an agent-based simulation. Each agent follows a simple pheromone-based rule to choose exits; the pheromone field is updated to encourage exploitation of good routes. This approach incorporates emergent coordination (no central planner required) and adapts to congestion in real time. However, it remains fundamentally a simulation: it provides no guarantee that the resulting routes are optimal, feasible, or compliant with external constraints. Similar simulation-centric frameworks reported by Pan et al. [71], Zheng et al. [106], Pelechano and Badler [74], Cao et al. [14], and more recently Bellas et al. [9] reach analogous conclusions: high-fidelity agent dynamics deliver descriptive realism but do not yield a prescriptive, optimality-preserving planner.

Later work by Bi [13], Tanaka et al. [87], and Agnihotri et al. [1] extended agent-based simulation with machine learning components. Tanaka et al. [87] used reinforcement learning to learn individual agent policies within a simulation; Agnihotri et al. [1] combined genetic algorithms with simulation to optimise global evacuation parameters. These methods are data-hungry and computationally expensive; they do not scale to large buildings or real-time planning scenarios.

The strength of simulation-based approaches is their expressiveness: they can model heterogeneous agents, stochastic behaviour, and complex spatial dynamics. The weakness is their opacity and scalability. A simulation that evacuates a 500-occupant building may take hours to converge. Moreover, simulation-based methods answer the question “what does evacuation look like?” rather than “what is the optimal plan?”. They provide post-hoc analysis but not online guidance.

Contemporary work (e.g., Mirahadi and McCabe [65] and Wang et al. [94]) combines simulation with data-driven methods: build a simulator, train a machine learning model on simulated or real evacuation data, then deploy the trained model as a fast surrogate for expensive simulation. This is a pragmatic compromise, but it does not address the semantic constraint problem or integrate formal reasoning about feasibility.

2.2.3 Agent-Based and Multi-Agent Models

Agent-based models (ABMs) are generative models in which simple local rules for each agent give rise to emergent global patterns. In evacuation contexts, ABMs model occupants as autonomous entities that sense the environment, apply local decision heuristics (move towards exit, avoid obstacles, follow social influence), and interact with other agents and the environment. Early work by Shi et al. [81] and Nishinari et al. [69] established the paradigm; modern extensions are surveyed by Yang et al. [99], Li et al. [54], and Yazdani and Haghani [100].

ABMs offer several advantages. They naturally model heterogeneous agents (children, elderly, mobility-impaired) with different speeds and behavioural rules. They capture emergent phenomena such as congestion, herding, and panic. They are inherently stochastic, reflecting the uncertainty in human behaviour. And they can incorporate spatial realism through continuous coordinates or fine-grained grid worlds.

However, ABMs have critical limitations for online planning. First, they are *descriptive*, not *prescriptive*. An ABM answers the question "given these behavioural rules, what happens?"; it does not answer "what plan minimises evacuation time?". To use an ABM as a planner, one must either run the simulation offline to explore different scenarios (expensive) or train a surrogate learned model on simulation data. Second, ABMs do not guarantee safety or compliance. An agent following a simple local heuristic may end up in a dead-end or violate a policy constraint. Third, ABMs do not scale well to real-time decision-making in large, complex buildings. Running an agent-based simulation of a 1000-occupant hospital in real time is infeasible.

Recent work by Li et al. [57] and Jafarian et al. [45] on multi-agent collaboration and dynamic routing introduces communication and feedback mechanisms, improving coordination, as do the evacuation-focused reinforcement learning pipelines of Ronchi and Nilsson [76] and the stochastic egress models of Cimellaro et al. [18]. However, these remain simulation-based approaches, not formal planning frameworks.

Gap 1: Simulation and ABM literature provides no principled method to encode semantic constraints (building codes, access policies, risk-based routing) into agent behaviour. Constraints are either hard-coded as agent heuristics (brittle and non-transferable) or ignored (unsafe).

Section synthesis. The evacuation planning literature reviewed in this section operates within two paradigms,graph-based optimisation and agent-based simulation,that fail for structurally different but complementary reasons. Graph-based planners possess the formal machinery to guarantee optimal paths (admissible heuristic search over a weighted graph), but they cannot maintain cost correctness: edge costs are static, hand-crafted, and never updated to reflect the congestion increases, edge invalidations, and node deletions that characterise a live evacuation. Agent-based models can track these dynamic changes at the individual-occupant level, but their local decision rules have no global cost function and therefore cannot reason about optimality, admissibility, or formal constraint satisfaction. Crucially, neither paradigm incorporates a semantic layer that defines which transitions are valid independently of cost,a building ontology that restricts the feasible successor set based on occupant type, hazard status, and regulatory policy (Section 2.3). Without such a layer, the planner cannot distinguish a geometrically short route that violates fire code from a compliant route that is slightly longer. The result is a three-way gap: the evacuation literature lacks (i) a formal constraint system to define transition validity, (ii) a cost model that tracks dynamic edge-weight changes, and (iii) a search framework that exploits both. These three requirements correspond, respectively, to the knowledge representation, machine learning, and heuristic search paradigms reviewed in the following sections.

2.3 Knowledge Representation for Built Environments

2.3.1 Ontologies for Building Modelling

An ontology in the knowledge representation sense is a formal specification of the concepts, properties, and relations in a domain. In the context of buildings and indoor environments, ontologies capture the structural, functional, and semantic properties of spaces, objects, and actors.

Early work by Kim et al. [49] and Onorati et al. [70] applied OWL-based ontologies to indoor navigation and emergency management. These works recognised that a geometric model of the building (coordinates, floor plans) is necessary but insufficient for evacuation planning. An ontology layer overlays semantic types (room, corridor, exit), access policies, and hazard information. Kim et al. [49]

developed an ontology for emergency evacuation that distinguished between administrative spaces, circulation spaces, and exits, and encoded evacuation-specific rules. Onorati et al. [70] proposed a more expressive ontology-based modelling approach for indoor environments, with explicit handling of accessibility constraints.

More recent work by Daher et al. [20] extended these approaches to include time-dependent aspects and real-time sensor integration. The ontology is no longer static: as sensors report fire, congestion, or structural damage, these facts are asserted into the ontology, and the reasoner refreshes the set of valid evacuation routes. Parallel contributions by Tashakkori et al. [89], Vanclooster et al. [91], and Nikoohemat et al. [68] confirm the feasibility of spatiotemporal semantic reasoning for emergency response but typically stop short of coupling their ontologies to a dynamic search procedure.

The strength of ontology-based approaches is their formal expressiveness and reusability. An OWL ontology defines a class hierarchy and constraints that can be checked by automated reasoners. A transition rule expressed in SWRL is transparent, auditable, and transferable to other buildings with the same structure. An ontology is also *separable* from the planner: one can modify the ontology to encode new constraints without changing the planning algorithm.

However, the literature in this area is limited and fragmented. Most ontology work is proof-of-concept: a small building is modelled, a toy planning problem is solved, and the approach is not evaluated against classical baselines or scaled to realistic buildings. Moreover, existing ontologies do not integrate cost models: they specify which transitions are feasible but not their cost. This creates a gap between knowledge representation and planning: an ontology tells you which routes are valid, but not which valid route is best.

Daher et al. [20] addressed this partially by incorporating cost data properties into the ontology, but the costs are static and hand-assigned. No work in the literature couples an ontology with a learned cost model.

Gap 2: Existing building ontologies are not integrated with planning algorithms. There is no standard approach for a planner to query an ontology for valid transitions and then search over them using modern heuristic algorithms.

To clarify the planning implications: an ontology, when equipped with SWRL transition rules, functions as a *constraint system* that partitions the set of all possible transitions into feasible and infeasible subsets. For any state n and

candidate successor n', the ontology determines whether the transition (n, n') belongs to the feasible successor set $\text{Succ}_{\mathcal{O}}(n)$,for instance, by checking occupant mobility class against staircase accessibility, or by verifying that the connecting door has not been flagged as blocked by a hazard assertion. This is a *feasibility* judgement, not an *optimality* judgement: the ontology can assert that a transition is valid, but it cannot assert that it is cheap, fast, or preferable. Computing the optimal path among all feasible transitions requires a cost function $c(n, n')$ and a search algorithm that minimises cumulative cost subject to the ontology's feasibility constraints,a role that the ontology itself is not designed to fill. In the terminology of the search framework reviewed in Section 2.4, the ontology defines the *graph* (which nodes and edges exist and which are currently valid), but it does not define the *edge weights* (how costly each valid transition is) or the *heuristic* (how far each state is from the goal). An ontology is therefore a necessary but insufficient component of an evacuation planner: it prevents the planner from producing transitions outside the feasible successor set, but it cannot, on its own, rank the transitions that remain.

2.3.2 OWL and SWRL in Spatial Reasoning

OWL (Web Ontology Language) is a description logic-based language for defining ontologies. It provides class hierarchies, object and data properties with domain and range restrictions, and support for class restrictions and property chains [7, 67]. SWRL (Semantic Web Rule Language), formalised by Horrocks et al. [42], extends OWL with Horn-clause rules, enabling the expression of more complex constraints and inferences. Probabilistic extensions such as those of Bellodi and Riguzzi [10] further allow uncertainty to be modelled over these rule bases.

In the spatial reasoning literature, OWL and SWRL have been applied to robot navigation, path planning, and geographic information systems. Kim et al. [49] used OWL to represent building topology and SWRL to encode movement rules (e.g., "a person in a room can exit to a corridor if there is an open door"). The rules are executed by a reasoner, which derives all valid transitions.

The advantage of this approach is automation: the reasoner performs inference over the ontology, deriving consequences that would be tedious to enumerate manually. However, reasoning over OWL and SWRL is computationally expensive. In the worst case, reasoning is undecidable (if full first-order logic features are used); in practice, restricted fragments (such as OWL 2 DL) are decidable but

may require exponential time. For real-time planning, this is a concern.

Moreover, OWL and SWRL are designed for expressing static knowledge and global constraints, not for efficient state-space search. A planner cannot simply invoke the reasoner at each step; instead, the reasoner must be queried to determine which transitions are valid, and the planner must handle the search logic separately.

Gap 3: The integration between ontological reasoning and state-space search is underspecified. Literature does not provide a clear algorithm for how a planner queries an ontology at each step, nor does it address the computational cost of repeated reasoning over a dynamic ontology.

2.3.3 BIM-to-Ontology Pipelines

Building Information Modelling (BIM) is a digital representation of a building's physical and functional characteristics. A BIM model captures geometry, materials, systems, and occupancy information in a structured format (typically IFC,Industry Foundation Classes). BIM is widely used in architecture, engineering, and facility management.

A natural question is whether BIM data can be automatically converted into a semantic ontology suitable for planning. Kim et al. [49] and Daher et al. [20] explored BIM-to-ontology conversion pipelines, and complementary work by Pauwels et al. [72], Zhu and Wu [107], Kang and Li [46], and Isikdag et al. [44] has defined standards and lifting rules that map IFC and IndoorGML representations into OWL. The pipeline typically extracts space definitions from the BIM model, infers connectivity from geometric adjacency, assigns classes based on space function, and populates an ontology.

The advantage is automation: if the building is already modelled in BIM (as is increasingly common), ontology construction becomes a data transformation task, not a manual knowledge engineering effort. This improves scalability and reproducibility.

However, BIM-to-ontology pipelines face several challenges. First, BIM models encode geometric and functional information but not semantic categories relevant to evacuation (restricted-access zones, evacuation assembly points, hazard areas). These must be added manually or inferred from heuristics. Second, BIM models are designed for architects and engineers, not for automated reasoning; the ontological interpretation of a BIM element may be ambiguous. Third, BIM update frequency

is low; once a building is handed over to facility management, BIM is rarely updated. Real-time sensor data and changing conditions are not reflected in the BIM model.

Existing work does not address these challenges systematically. The result is that BIM-to-ontology pipelines are promising but immature, suitable for offline planning but not for real-time responsive systems.

Gap 4: No integrated BIM-to-ontology-to-planner pipeline exists in the literature. Each component (BIM extraction, ontology construction, planning) is studied in isolation.

Section synthesis. The knowledge representation literature reviewed in this section establishes that ontologies can formally encode the semantic structure of a building,spatial topology, occupant classes, accessibility policies, and hazard status,and that SWRL rules can derive the feasible successor set at each state. However, the paradigm fails at the planning boundary for two structural reasons. First, the ontology defines transition validity but not transition cost: it can assert that Corridor_3 $\rightarrow$ Stairwell_B is a feasible transition for an able-bodied adult, but it cannot assert that this transition costs 12 seconds under current congestion. Cost estimation requires either hand-crafted assignment (static, fragile, and unable to track dynamic edge-weight changes during a live evacuation) or a learned model trained on historical or simulated data,precisely the machine learning paradigm reviewed in Section 2.5. Second, the ontology is a knowledge base, not a planner: it can answer the query "which transitions are valid from state n?" but it cannot answer "which sequence of valid transitions minimises cumulative cost to the goal?". Answering the latter requires a search algorithm (Section 2.4) that traverses the ontology-defined graph, evaluates transition costs, and applies an admissible heuristic to prune the search space. The ontology thus occupies a precise and indispensable role,it is the constraint layer that restricts the search space to feasible transitions,but it must be coupled with both a cost model and a search procedure to produce an actionable evacuation plan. No work reviewed in this section achieves that coupling.

2.4 Heuristic Search Algorithms for Dynamic Environments

2.4.1 A* and Its Variants

The A* algorithm, introduced by Hart et al. [34], is the foundation of modern heuristic search, building on earlier work by Pohl [75] and systematised by Pearl [73]. A* maintains an open list of candidate nodes, prioritises by $f(n) = g(n)+h(n)$ where $g(n)$ is the cost-to-reach and $h(n)$ is a heuristic estimate of cost-to-goal, and expands nodes in best-first order. If $h(n)$ is admissible (never overestimates true cost), A* is guaranteed to find an optimal solution. Abstraction-based heuristic construction, surveyed by Holte [41], provides a principled way to derive admissible heuristics from problem structure, a technique relevant to the ontology-derived heuristic developed in Chapter 4.

A* has been applied to evacuation planning (e.g., Wang et al. [93], Wang et al. [95]). In this context, nodes represent spatial situations and edges represent transitions (moving through a door, traversing a corridor, descending a staircase). Edge costs are traversal costs, and the heuristic is typically geometric distance to the nearest exit.

The strength of A* is its optimality guarantee and efficiency relative to uninformed search. The weakness is its assumption of a static environment. If the graph changes (an edge is deleted, a cost is updated), the prior search tree is invalidated, and A* must restart from scratch.

In evacuation contexts, the static environment assumption is violated constantly: a fire blocks an exit, a corridor becomes obstructed, new occupancy data arrives. For this reason, classical A* is insufficient for real-time dynamic planning.

The dependence on two external inputs,the graph structure and the edge costs,makes A* a *consumer* of information that it cannot itself produce. The graph structure, in an evacuation planner, must encode which transitions are currently valid: which doors are open, which corridors are passable, which staircases are accessible to a given occupant class. This is precisely the role of the ontology-guided constraint system reviewed in Section 2.3; without such a system, the search algorithm operates on a graph that may contain edges corresponding to transitions not in the ontology-defined feasible successor set,a smoke-logged corridor still treated as traversable, or a fire-rated door whose status has changed but whose edge persists in the graph. The edge costs, conversely, must reflect the actual traversal time or risk of each valid transition under current conditions,congestion-dependent corridor costs, hazard-proximity penalties, capacity-limited bottleneck delays. Hand-crafted costs (Section 2.2.1) are static and therefore incorrect as soon as

occupant density shifts or a hazard propagates. Learned costs (Section 2.5) can in principle track these changes, but they introduce their own failure modes: the learned cost $\hat{c}(n, n')$ may overestimate (destroying admissibility and with it the optimality guarantee) or underestimate (directing evacuees toward high-cost or invalidated edges). The correctness of A*'s output is therefore bounded by the correctness of its inputs, and in evacuation planning, neither input,graph structure nor edge costs,is currently supplied by a system that guarantees real-time accuracy.

2.4.2 Incremental Search: LPA* and D* Lite

To address the limitations of A* in dynamic environments, several algorithms have been proposed for incremental search: reusing prior computation when the graph changes, rather than restarting from scratch.

The Lifelong Planning A* (LPA*) algorithm, introduced by Koenig et al. [52], maintains consistency between a previous search and the current search when edge costs change. When the cost of an edge decreases, LPA* identifies affected nodes, updates their costs, and only re-expands nodes whose $h(n)$ values have changed. This can be significantly faster than restarting A* from scratch, especially if the change is localised.

The D* and D* Lite algorithms, introduced by Stentz [82], Stentz [83], and Koenig and Likhachev [51], extend this approach. D* Lite is more efficient than LPA* because it expands nodes in backward search (from goal towards start) and benefits from tighter heuristics. When edge cost updates or transition invalidation occur, D* Lite re-expands only affected nodes. Related anytime incremental variants including ARA* [58], AD* [59], and multi-heuristic A* [4] demonstrate that incremental and anytime search can be combined to handle both sub-optimality bounds and cost updates within a single framework.

The theoretical advantage of incremental search is dramatic: in a graph with N_{nodes} nodes, A* requires $O(N_{\text{nodes}} \log N_{\text{nodes}})$ time in the worst case; incremental algorithms require $O(k \log N_{\text{nodes}})$ where k is the number of affected nodes. If the change is localised (e.g., a single exit blocked, a single corridor becomes congested), $k \ll N_{\text{nodes}}$ and incremental search is much faster.

However, there is an important caveat. Incremental algorithms assume that the prior plan is valid and that the goal is fixed. If the goal changes (e.g., the nearest exit becomes blocked and the evacuee must re-route to a different exit), the algorithm must restart. Moreover, incremental algorithms assume that edge costs

are known at query time. If costs are learned from data and updated as new data arrives, the incremental algorithm must account for this.

The evacuation literature has not widely adopted incremental search. Most work applies A* or other uninformed methods. Ferguson et al. [24] surveyed path planning algorithms in robotics and noted that incremental search is valuable for dynamic environments; this insight has not been systematically transferred to evacuation planning. One exception is the incremental indoor path planner of Liu et al. [61], but it does not couple incremental search to a formal ontological transition model.

Gap 5: Incremental search algorithms are underutilised in evacuation planning. There is no established practice for integrating LPA* or D* Lite into evacuation planning systems, nor clear guidance on when incremental search outperforms A*.

2.4.3 Comparison of Re-planning Strategies

When a change in the environment makes the current plan invalid, the planner must re-plan. There are three strategies: re-planning from scratch, anytime re-planning, and incremental re-planning.

Re-planning from scratch: Discard the old plan and invoke A* again. This is guaranteed to find an optimal plan but is slow.

Anytime re-planning: Keep the old plan and return it immediately, while computing a new plan in the background. This minimises latency but may result in suboptimal routing. Tang et al. [88] surveyed anytime algorithms; Hassan and Tucker [36] applied them to evacuation.

Incremental re-planning: Use an algorithm like LPA* or D* Lite to reuse computation from the previous plan. This balances optimality and latency.

The evacuation literature has not performed systematic comparisons of these strategies. Zhang et al. [104], Zhai and Feng [103], and Yan and Shu [98] study variants of dynamic routing but do not compare re-planning algorithms. The result is a lack of guidance for practitioners on when each strategy is appropriate.

Moreover, the choice of re-planning strategy is orthogonal to the choice of cost model. Whether costs are hand-assigned or learned, the planner must still choose a re-planning strategy. This choice is treated as engineering, not research, and is thus underexplored.

Gap 6: There is no principled comparison of re-planning strategies in the evacuation context. The interaction between learned costs and incremental search is unexplored.

Section synthesis. Heuristic search provides the formal machinery that the evacuation paradigms reviewed in Section 2.2 lack: a principled algorithm that, given correct inputs, returns an optimal plan with provable guarantees. Yet the guarantee is conditional,admissibility of $h(n)$ and correctness of $c(n, n')$,and the search algorithm cannot verify or produce either condition on its own. The graph over which search operates must reflect the current set of valid transitions, which changes whenever a hazard invalidates an edge (fire engulfs a corridor) or a policy constraint removes a node from the feasible successor set of a particular occupant class (a wheelchair user reclassified to an alternative exit). Maintaining this graph in real time is a knowledge representation problem (Section 2.3), not a search problem. The edge costs that drive node-expansion ordering must reflect current traversal conditions,congestion-dependent delays, hazard-proximity penalties,which change asynchronously during an active evacuation. Producing accurate, up-to-date costs is a learning problem (Section 2.5), not a search problem. Heuristic search thus occupies the centre of the evacuation planning pipeline: it is the only component that can guarantee optimality, but it depends on the ontology to supply a correct graph and on a learned model to supply correct costs. Without both, its guarantees are formal but ungrounded. This three-way dependency,ontology for transition validity, learned model for transition cost, search for optimal sequencing,cannot be satisfied by any single paradigm and motivates the integration architecture examined in Section 2.6.

2.5 Machine Learning for Evacuation and Navigation

2.5.1 Sequence Models in Spatial Prediction

A trajectory through an evacuation is a sequence of spatial transitions: person at room A → corridor B → vestibule C → staircase D → exit E. Predicting the next transition given the history of transitions is a sequence modelling problem.

Sequence models have been successfully applied to trajectory prediction in pedestrian dynamics (Li et al. [54]), crowd simulation (Khalilpourazari and Pasandideh [48]), and navigation (Sun et al. [84]). The standard approach is to train a recurrent neural network or Transformer [92] on historical trajectory data, learning the conditional probability $P(\text{next spatial state} \mid \text{history of states})$. Representative examples include Social LSTM [5], Social GAN [33], Trajectron++ [77], the MDST-DGCN graph model of Liu et al. [60], the indoor goal-oriented GPVP-transformer of He et al. [37], and the CoordConv-based pedestrian trajectory model of Wu and Chen [97]. Given a starting location and a destination, the model predicts the most likely intermediate states. A parallel line of work has explored generative adversarial imitation learning for pedestrian evacuation dynamics, though published results in this area remain limited.

Sequence models have two attractive properties. First, they can capture long-range dependencies: the optimal next step may depend on states many steps in the past (e.g., whether a person entered the building from the north or south may influence which exit they are heading towards). Second, they naturally quantify uncertainty: the model outputs a probability distribution over next states, not a point prediction.

However, sequence models for trajectory prediction have critical limitations for planning. First, they are *descriptive*, not *prescriptive*. A trained model predicts what an evacuee is likely to do, not what they should do to optimise evacuation time or compliance. To use such a model as a planner, one must either assume that typical behaviour is optimal (often false under stress) or train the model with reward signals indicating optimal behaviour (which requires labelled data or simulation).

Second, sequence models are typically trained on offline data. If the training data were collected under different conditions (different building layout, different population, different initial conditions), the model may not generalise. Sun et al. [84] and Huang et al. [43] acknowledge this but do not provide systematic solutions.

Third, sequence models provide no guarantees on feasibility or compliance. A predicted trajectory may violate building codes, accessibility constraints, or policy requirements. Post-processing (filtering invalid trajectories) can help but is ad hoc.

These limitations expose a fundamental incompatibility between sequence models and the heuristic search framework reviewed in Section 2.4. A search algorithm such as A* requires two inputs: a cost function $c(n, n')$ for each transition and

an admissible heuristic $h(n)$ that never overestimates the true cost-to-goal. Sequence models supply neither. They optimise a likelihood objective,typically the cross-entropy between predicted and observed transition distributions,which is a statistical measure of fit, not a planning cost. A trajectory that maximises likelihood under the training distribution maximises the probability of reproducing *typical* evacuee behaviour; it does not minimise evacuation time, exposure to hazard, or any other planning-relevant cost. In a dynamic evacuation, congestion increases edge traversal costs (higher occupant density raises the time to traverse a corridor segment), fire propagation invalidates edges and nodes entirely (a corridor engulfed by flashover is removed from the feasible graph), and structural failure deletes edges without warning (a collapsed ceiling blocks a previously viable route). These changes are continuous and asynchronous, meaning the cost function $c(n, n')$ that the search algorithm operates on shifts between planning cycles. A sequence model trained on a static historical distribution cannot track these shifts; the divergence between the likelihood-optimal trajectory and the cost-optimal plan grows with every unobserved edge-cost change, and can become arbitrarily large: the most probable corridor under the training distribution may carry the highest traversal cost,or may no longer exist,under the current situation.

Furthermore, the constraint enforcement mechanisms available in ontology-based representations (Section 2.3) have no counterpart in sequence model inference. An ontology equipped with SWRL transition rules restricts the successor set at each state to transitions that are semantically valid,a wheelchair user cannot take the stairs, a fire-rated door cannot be held open. Sequence models, by contrast, assign non-zero probability to every transition in the vocabulary, including transitions not in the ontology-defined feasible successor set,a mobility-impaired evacuee directed toward a staircase, a route through a fire-rated door whose hold-open device has been released, a path traversing a corridor whose edge has been deleted from the graph due to smoke logging. The model has no internal mechanism to enforce $P(\text{transition} \notin \text{Succ}_{\mathcal{O}}(n)) = 0$; it can only learn to assign low probability to such transitions if they appear rarely in training data. Under distribution shift,an emergency type, building configuration, or occupant profile not represented in training,the model may assign high probability to transitions outside the ontology-defined feasible successor set precisely when constraint enforcement is most critical: during the novel, high-consequence situations for which the evacuation planner exists.

Gap 7: Machine learning trajectory models are decoupled from knowledge representations of feasibility and compliance. There is no integrated framework

that uses learned transition probabilities while respecting ontological constraints.

2.5.2 LSTM and BiLSTM Architectures

The Long Short-Term Memory (LSTM) architecture, introduced by Hochreiter and Schmidhuber [40] and subsequently analysed at depth by Greff et al. [31], is a type of recurrent neural network designed to learn long-range dependencies in sequences. An LSTM cell maintains a cell state (memory) that is updated selectively via gate mechanisms, allowing information to flow over many time steps without vanishing or exploding. The gated recurrent unit variant of Cho et al. [16] offers a lighter-weight alternative that is often competitive for moderate-length sequences.

The Bidirectional LSTM (BiLSTM), introduced by Schuster and Paliwal [78], extends LSTM by processing the sequence in both forward and backward directions, concatenating the outputs. This is particularly effective for tasks where the context of an element depends on both past and future elements.

In the evacuation and navigation literature, LSTMs and BiLSTMs have been applied to trajectory prediction and crowd simulation. Graves and Schmidhuber [30] used LSTM for sequence labelling; Mirahadi and McCabe [65], Wang et al. [94], and more recently Wang et al. [93] applied LSTMs to real-time evacuation prediction. Hassaan et al. [35] and Agnihotri et al. [1] used LSTMs within agent-based simulations to learn individual agent policies.

The strength of LSTM/BiLSTM is their capacity to model non-linear, long-range temporal dependencies. A person's evacuation behaviour depends on where they came from, where others are moving, and where exits are located; LSTMs can learn these dependencies from data.

However, LSTMs have several limitations for evacuation planning. First, they are *learned models*, not *interpretable models*. It is difficult to audit whether an LSTM has learned physically plausible or policy-compliant routing. Second, LSTMs require substantial training data. For rare scenarios (building types, population distributions, emergency types) data may not exist. Third, LSTMs do not provide optimality guarantees. An LSTM can predict that a person will move to location X next, but this does not mean X is the best next location for evacuation time.

Recent work by the thesis authors (Djemai et al. [23]) applies BiLSTM to learn cost functions for spatial transitions in evacuation. This is closer to the planning

use case: rather than using BiLSTM to predict trajectories directly, use it to learn edge costs that are then fed into a planner. However, this work does not integrate with ontological constraints or discuss incremental re-planning.

The distinction between using a BiLSTM to predict trajectories and using it to estimate edge costs reveals a deeper tension. While the cost-estimation framing aligns with the input requirements of A* and LPA* (Section 2.4), it introduces an optimisation mismatch that the search literature does not address. The BiLSTM is trained via maximum likelihood on observed transitions, meaning the learned cost $\hat{c}(n, n')$ reflects the frequency of a transition under the training distribution, not the true traversal cost under the current evacuation situation. The consequences depend on the direction of error. If the learned cost *underestimates* the true traversal cost of an edge,because, for example, the training data did not include high-congestion corridor segments where occupant density doubles the traversal time, or post-flashover conditions where a corridor's edge weight should be infinite (deleted edge),then the search algorithm expands nodes along routes that appear cheap but are in fact costly or impassable; the resulting plan is feasible only by accident and may route evacuees into danger. If the learned cost *overestimates*, the derived heuristic $h(n) = \sum \hat{c}$ exceeds the true cost-to-goal $h^*(n)$, violating the admissibility condition $h(n) \leq h^*(n)$; A* may then prune the optimal path from the search tree and return a suboptimal plan (Section 2.4.1). In evacuation, suboptimality translates directly to excess evacuation time,additional seconds of exposure to smoke, heat, and structural risk for every evacuee on the affected route. Furthermore, in an incremental search framework (Section 2.4.2), cost updates trigger local re-expansion of affected nodes; if the learned cost function updates its estimates after each batch of sensor data, the number of affected nodes k may approach N_{nodes} at each re-planning cycle, eliminating the efficiency advantage of incremental search entirely. No existing work quantifies this interaction between learned cost volatility and incremental search performance in evacuation settings.

Gap 8: BiLSTM and other neural sequence models are not integrated with formal planning frameworks. The use of learned costs within heuristic search is addressed in isolation from ontological constraint satisfaction.

2.5.3 Neural Heuristic Learning

A heuristic function $h(n)$ estimates the cost from state n to a goal state. An admissible heuristic never overestimates; this is essential for A* to guarantee

optimality. Traditionally, heuristics are hand-crafted (e.g., straight-line distance, Manhattan distance).

Neural heuristics are heuristics learned from data. The idea is to train a neural network on instances of the planning problem, learning a function that maps states to cost estimates. If the training set is large and representative, the learned heuristic may generalise to new problem instances.

The challenge with neural heuristics is ensuring admissibility. A neural network has no intrinsic reason to respect the constraint $h(n) \leq \text{true cost}(n)$. If the learned heuristic overestimates, A* may return a suboptimal solution. Representative efforts to bridge learning and search include the Value Iteration Network of Tamar et al. [86], the imitation-learning planner of Choudhury et al. [17], the Neural A* architecture of Yonetani et al. [101], the deep reinforcement learning search of Agostinelli et al. [2], and the classical-planning heuristic learner of Shen et al. [80]; training of these models is commonly performed with the Adam optimiser of Kingma and Ba [50]. Wang et al. [96] further combines graph neural networks with heuristic search for real-time building evacuation. Despite these advances, the problem of guaranteeing admissibility of a learned heuristic remains open.

In evacuation contexts, neural heuristics have been proposed but not widely studied. The thesis's BiLSTM work (Djemai et al. [23]) can be understood as learning a neural cost function rather than a heuristic, but the distinction is blurry: a cost function is an estimate of the immediate cost of an action, while a heuristic is an estimate of the cumulative future cost. The relationship between these is underexplored.

The structural difficulty is that admissibility is a worst-case guarantee,$h(n) \leq h^*(n)$ must hold for every reachable state n,while neural networks are trained to minimise average-case loss over a finite dataset. A neural heuristic that achieves low mean-squared error on the training distribution can still overestimate,and therefore violate admissibility,on out-of-distribution states. In evacuation planning, these out-of-distribution states correspond precisely to the rare, high-consequence situations where optimality matters most: flashover propagation that deletes multiple corridor edges simultaneously, concurrent exit blockages that restructure the feasible graph topology, or mass panic-induced bottlenecks that multiply edge traversal costs by an order of magnitude. While Yonetani et al. [101] and Agostinelli et al. [2] demonstrate empirical gains in static domains, evacuation environments violate their key assumption: that the cost landscape is stationary between training and deployment. In a dynamic evacuation, each new sensor

reading triggers concrete graph modifications: a smoke detector activation deletes a corridor edge, a congestion sensor multiplies an edge's traversal cost, a structural alert removes an entire node (room) and all its incident edges. The neural heuristic, frozen at training time, cannot track these shifts, and no mechanism exists to re-calibrate it within the latency budget of a real-time planner. This contrasts sharply with the ontological constraint system of Section 2.3, which can be updated in real time via assertion of new facts,`hasStatus(Corridor_3, Blocked)` deletes the corresponding edge from the ontology-defined feasible successor set within a single reasoning cycle, immediately restricting the search space. The neural heuristic, by contrast, encodes the cost landscape in a continuous weight matrix that cannot incorporate a discrete edge deletion or node removal without retraining or fine-tuning,a process that is orders of magnitude too slow for the sub-second re-planning cycles required during active evacuation.

Gap 9: There is no established theory or practice for using neural heuristics in evacuation planning. The admissibility of neural heuristics and their integration with incremental search is unclear.

Section synthesis. The three subsections above reveal a cumulative structural failure of the machine learning paradigm when applied to evacuation planning. Sequence models (Section 2.5.1) optimise a likelihood objective that diverges from planning cost; they produce descriptive predictions, not prescriptive plans, and provide no mechanism to enforce the hard safety constraints that an ontology (Section 2.3) can express. LSTM and BiLSTM architectures (Section 2.5.2) bring the learned model closer to the search interface by producing edge-cost estimates, but those estimates inherit the training distribution's biases and offer no admissibility guarantee,the very property on which A* and LPA* (Section 2.4) depend for optimality. Neural heuristic learning (Section 2.5.3) attacks the admissibility problem directly but cannot solve it within a purely statistical framework: worst-case bounds on a neural function approximator cannot be certified from finite training data, and the dynamic, non-stationary cost landscape of an active evacuation invalidates any static training-time calibration. Each paradigm fails for a structurally different reason,wrong objective, wrong guarantees, wrong constraint mechanism,but the failures are connected: they all stem from the absence of a formal interface between what the learning component produces (continuous, probabilistic, average-case outputs) and what the planning component requires (discrete, constraint-satisfying, worst-case-bounded inputs). This structural mismatch cannot be resolved from either side alone: improving the ML method does

not introduce the hard constraint enforcement and admissibility guarantees that the planning side requires, while the symbolic ontology and search framework reviewed in Sections 2.3 and 2.4 cannot, on their own, generate the data-driven cost estimates needed to rank transitions in novel or partially observed evacuation situations. Neither pure ML nor pure symbolic reasoning is sufficient; the mismatch demands an integration architecture in which each component compensates for the formal limitations of the other,a challenge taken up by the neuro-symbolic literature reviewed next.

2.6 Neuro-Symbolic AI

2.6.1 Architectures and Integration Strategies

Neuro-symbolic AI seeks to combine the learning capabilities of neural networks with the reasoning and interpretability of symbolic systems. Neither pure learning nor pure reasoning alone is sufficient for complex tasks: learning without constraints is data-hungry and opaque; reasoning without learned models is brittle and inflexible.

Garcez and Lamb [25], Garcez and Lamb [26], Hitzler and Sarker [39], Kautz [47], Besold et al. [12], and Marra et al. [64] survey neuro-symbolic approaches. The landscape is diverse, encompassing several architectural patterns, with concrete systems such as DeepProbLog [62] exemplifying tight coupling between neural predictors and logical rule bases.

Sequential integration: Train a neural network on data, then use its outputs to constrain or guide a symbolic solver. For example, a neural network might predict an emergency state (fire in zone A) and feed this into an ontological reasoner to compute ontology-constrained transition sequences. The network and reasoner are decoupled; information flows one direction.

Embedded reasoning: Embed constraint-satisfaction logic into a neural network, so that the network's outputs respect a priori constraints by construction. For example, a neural network can be constrained to output a valid state (one that satisfies all ontological rules) rather than an arbitrary output. This requires careful architecture design and is computationally more expensive.

Hybrid search: Use a symbolic planner that leverages learned components. For

example, invoke a learned model to estimate edge costs, feed these into A*, and trust that the planner will find a good path despite the costs being approximate. This is pragmatic and computationally tractable, but lacks formal guarantees.

The evacuation and planning literature has begun to explore neuro-symbolic approaches. Khalilpourazari and Pasandideh [48] combined learning and optimisation for emergency response. Li et al. [56] applied hybrid methods to real-time routing. Recent surveys by Leofante et al. [53] and Confalonieri et al. [19] further map the design space for safety-critical neuro-symbolic decision making. However, these works do not explicitly frame their approach as neuro-symbolic nor do they provide a principled architecture for integrating learning and reasoning.

More broadly, neuro-symbolic AI for planning is an active research area in the robotics and automated planning communities, but evacuation planning is a relatively niche application. The application of neuro-symbolic techniques to evacuation has been limited.

Gap 10: There is no systematic study of neuro-symbolic architectures for evacuation planning. The interaction between learned cost models and symbolic constraints (expressed in ontologies) is underspecified.

When evaluated against the specific requirements of evacuation planning, each of the three integration patterns above exhibits a distinct structural limitation. Sequential integration decouples the learned component from the symbolic reasoner: the neural network predicts a hazard state or estimates edge costs, and these outputs are consumed by the ontological reasoner or search algorithm in a single forward pass. In an evacuation context, this means the learned cost function $\hat{c}(n, n')$ is fixed at inference time and cannot be corrected when the ontology detects that a predicted transition is not in the feasible successor set $\mathrm{Succ}_{\mathcal{O}}(n)$,for example, when a corridor that the learned model rates as low-cost has been removed from the graph by a fire-status assertion. The planner receives inconsistent inputs (a cost for an edge that no longer exists) and must either discard the learned cost entirely or accept a stale estimate. Embedded reasoning addresses this by enforcing ontological constraints within the neural architecture itself, but at a computational cost that scales with the number of constraints: in a building with hundreds of rooms, doors, and occupant-class-specific accessibility rules, embedding all SWRL-derived transition constraints into the network's forward pass introduces latency that conflicts with the sub-second re-planning cycles required during active evacuation. Hybrid search,feeding learned costs into A* or LPA*,avoids the latency of embedded reasoning and the staleness of sequential

integration, but inherits the admissibility problem analysed in Section 2.5.3: the learned heuristic may overestimate the cost-to-goal for states not represented in training, destroying the optimality guarantee of the search, or underestimate for transitions whose true cost has increased due to congestion or partial blockage, directing evacuees toward high-cost or invalidated edges. None of the three patterns resolves all three requirements simultaneously: real-time ontology-guided transition filtering, accurate learned edge costs, and admissible search guarantees.

2.6.2 Neuro-Symbolic Planning

Neuro-symbolic planning refers to planning systems that use both learned and symbolic components. A simple instance is: train a neural network to predict the next state given the current state and an action, then use these predictions as a learned model of the environment in a symbolic planner (PDDL-based, or graph-based).

More sophisticated neuro-symbolic planning systems use neural networks to learn value functions (expected future reward from a state) or policy functions (which action to take in a state), then incorporate these into a symbolic planner as heuristics or constraints.

The intersection of learning, planning, and symbolic reasoning is active in robotics,for example, the mapless navigation agent of Tai et al. [85] and the deep navigation architecture of Mirowski et al. [66],as well as in earlier case-based reasoning work [3]. In evacuation, however, these strands remain largely disjoint: existing work either uses learning (ML trajectory prediction) or planning (heuristic search) but rarely both in a principled, integrated manner.

One exception is the thesis's own prior work (Djemai et al. [21], Djemai et al. [22]). These papers developed an ontology-based domain model for evacuation and integrated it with planning algorithms. The BiLSTM cost learning work (Djemai et al. [23]) extends this by adding a learned cost component. However, the integration is sequential, not fully hybrid: the LSTM learns costs offline, then these costs are fed into the planner at runtime. A tighter integration,where learning and planning feedback one another,is not yet realised.

Gap 11: Neuro-symbolic planning systems for evacuation are immature. There is no established practice for tight integration of learned costs and symbolic planning with ontological constraints.

Gap 12: The use of learned costs within incremental re-planning algorithms (like LPA* or D* Lite) has not been studied in the context of evacuation.

Section synthesis. The neuro-symbolic literature offers architectural templates for combining learned and symbolic components, but none of the existing patterns satisfies the joint requirements of evacuation planning as identified across the preceding sections. The ontology (Section 2.3) must filter transitions in real time, restricting the search graph to the feasible successor set as hazard assertions invalidate edges and nodes. The learned model (Section 2.5) must supply edge costs that reflect current traversal conditions,congestion-dependent delays, hazard-proximity penalties,without violating the admissibility condition on which the search algorithm's optimality guarantee depends (Section 2.4). The search algorithm must consume both inputs,a dynamically filtered graph and dynamically updated costs,and return an optimal plan within the latency budget of a live evacuation. Each of the three neuro-symbolic integration patterns reviewed above satisfies at most two of these three requirements: sequential integration preserves search guarantees and ontological filtering but cannot correct stale learned costs; embedded reasoning enforces constraints within the learned component but at latency incompatible with re-planning; hybrid search combines learned costs with formal planning but provides no mechanism to enforce ontological transition validity. The gap is therefore not the absence of neuro-symbolic techniques *per se*, but the absence of an integration architecture that mediates all three layers,ontology-guided constraint enforcement, learned cost estimation, and admissible heuristic search,within a single, formally grounded planning loop. This is the specific architectural challenge that the gap analysis in Section 2.7 formalises and that the thesis addresses.

2.7 Gap Analysis and Research Positioning

Table 2.1 consolidates the surveyed literature against the capability dimensions that the earlier sections of this chapter identified as critical for indoor emergency evacuation: re-planning, indoor scope, dynamic routing, backtracking, mass evacuation, vertical navigation, ontological reasoning, explainability, and real-time event adaptation. Read column-wise, the table makes the structural nature of the gap explicit: optimisation, meta-heuristic, cellular-automata, and classical machine-learning approaches each satisfy only a narrow subset of the required

Table 2.1: Summary of prior related emergency evacuation studies and the open research gaps they leave. Symbols ✓, ✗, and ≈ mean Yes, No, and Partial, respectively.

Ref.	Data source	NoF	PP technique	RP	Indoor	DR	BT	Mass	VN	OR	XAI	RTEA
[27]	Synthetic + real city case	NA	MILP + robust optimisation	✗	✗	≈	✗	✓	✗	✗	✗	≈
[15]	Real road networks	NA	ACO-based multi-objective routing	✗	✗	✓	✗	✓	✗	✗	✗	✗
[54]	Hydrodynamic simulation + GIS	NA	CA + heuristic shortest path	✗	✗	✓	✗	✓	✗	✗	✗	≈
[63]	Synthetic + benchmark networks	NA	Fuzzy IP + improved ACO	✗	✗	✓	✗	✓	✗	✗	✗	≈
[45]	Behavioural simulation data	NA	Behaviour-aware routing model	✗	✗	≈	✗	✓	✗	✗	✗	✗
[57]	Multi-agent simulated networks	NA	Collaborative optimisation framework	≈	✗	✓	✗	✓	✗	✗	✗	≈
[87]	Yodogawa area of Osaka city	NA	CNN-assisted evacuation planning	✗	✗	✗	✗	✓	✗	✗	✗	≈
[84]	Post-hurricanes Katrina and Rita survey	NA	Enhanced logistic regression with decision trees	✗	✗	✗	✗	✓	✗	✗	✗	✗
[6]	San Francisco city seismic risks	NA	ANN-based site selection with shortest-path navigation	✗	✗	✓	✗	✓	✗	✗	✗	✗
[95]	Ship layout (grid-based)	1	Cellular automata + ACO	✗	✓	✗	✗	✓	✗	✗	✗	✗
[104]	Public hall experiments	1	Signage location optimisation	✗	✓	✗	✗	✓	✗	✗	✗	✗
[8]	Indoor robotic simulations	1	MPC-based planning	✓	✓	✓	≈	✗	✗	✗	✗	✓
[94]	Two quasi-emergency evacuation experiment videos	1	ML-based stepwise movement prediction	✗	✓	✗	✗	✗	✗	✗	✗	✗
[35]	German emergency floor plans	1	Faster region-CNN digitised evacuation	✗	✓	✗	✗	✗	✗	✗	✗	✗
[55]	3D GIS + fire simulation	8	A* + fire dynamics	✗	✓	✓	✗	✓	✓	✗	✗	≈
[105]	Mine network simulation	NA	FA-MDPSO + graph layout	✗	✓	✓	✗	✗	✓	✗	✗	✗
[102]	Connected sensor network	3	ML-based indoor AR navigation and emergency evacuation	✓	✓	✓	✗	✗	✓	✗	✗	✓
This thesis	Real building ontology	11	Neuro-Symbolic Semantic LPA* (BiLSTM-Guided)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Abbreviations: Number of floors (NoF); Path planning (PP); Re-planning (RP); Dynamic routing (DR); Backtracking (BT); Mass evacuation (Mass); Vertical navigation (VN); Ontological reasoning (OR); Explainability (XAI); Real-time event adaptation (RTEA).

capabilities, and none of the reviewed studies simultaneously supports ontological reasoning, backtracking, vertical multi-floor navigation, and real-time event adaptation in a single planner. This pattern directly substantiates the cluster-level gaps articulated in the remainder of Section 2.7 and motivates the neuro-symbolic positioning of this thesis, in which a formal building ontology is coupled with an incremental heuristic search and a learned guidance signal to close the capabilities that the prior literature leaves only partially addressed.

This section synthesises the gaps identified above and articulates the contribution of the thesis.

2.7.1 Synthesis of Identified Gaps

The literature review has identified twelve significant gaps. For brevity, we group them into four clusters:

1. **Semantic constraint encoding** (Gaps 1, 2, 4): Classical planning and simulation approaches do not integrate semantic domain knowledge (expressed as ontologies or constraints). Ontologies exist but are not coupled to planners. BIM-to-ontology pipelines are proof-of-concept. This gap directly relates to Challenge 1 in the research problem (Section 1.1), because geometric connectivity alone does not provide the semantic information required to distinguish ontology-constrained transitions or enforce building-specific routing constraints.

2. **Dynamic re-planning** (Gaps 5, 6, 12): Incremental search algorithms (LPA*, D* Lite) are underutilised in evacuation. Re-planning strategies are not compared systematically. The integration of learned costs with incremental search is unexplored. This gap directly relates to Challenge 2 (Section 1.1), because approaches that cannot efficiently reuse previous search effort when edge costs change or transitions become invalid cannot exploit the computational advantages required for responsive dynamic re-planning.

3. **Learned cost estimation** (Gaps 7, 8, 9): Machine learning models (LSTMs, neural heuristics) predict trajectories or learn costs but are decoupled from planning algorithms. Their integration with formal guarantees (admissibility, feasibility) is unclear. This gap directly relates to Challenge 3 (Section 1.1), because the use of learned guidance in this thesis requires a bounded integration mechanism that can improve search guidance without compromising

the admissibility properties of the underlying heuristic search.

4. **Neuro-symbolic integration** (Gaps 10, 11): Neuro-symbolic AI is theoretically appealing but is minimally applied to evacuation planning. The interaction between learned costs and symbolic constraints is underspecified. This gap relates to the overarching research problem (Section 1.1), because the combined treatment of semantic constraints, dynamic re-planning and learned guidance within a single formally bounded planning framework remains insufficiently addressed in the evacuation-planning literature reviewed in this chapter.

2.7.2 Thesis Positioning

The thesis addresses all four clusters by proposing an integrated framework for neuro-symbolic evacuation planning. Specifically:

1. **Chapter 3 – Ontological Domain Modelling:** Develops an OWL–SWRL ontology that formally encodes the building structure, situations, actions, and events. This ontology is designed to be reusable and separable from the planner, addressing Gap 2. The ontology also provides the basis for semantic heuristic derivation (addressing Gap 5).

2. **Chapter 4 – Heuristic Search with Re-planning:** Instantiates LPA* and D* Lite algorithms within the evacuation context, using the semantic heuristic derived from the ontology. This addresses Gaps 5 and 6 by demonstrating when incremental search outperforms from-scratch re-planning and providing a comparison of re-planning strategies.

3. **Chapter 5 – BiLSTM-Learned Cost Estimation:** Applies BiLSTM to learn transition costs from evacuation data, then integrates these learned costs into the heuristic search framework. This addresses Gaps 8, 9, and 12 by showing how neural costs can be safely incorporated into incremental search.

4. **Chapter 6 – Empirical Evaluation:** Evaluates the integrated neuro-symbolic framework on three structurally diverse buildings (Tower, Hospital, School), demonstrating both the expressiveness of the ontology and the practical advantages of learned costs and incremental re-planning.

The unifying theme is *integration*: rather than applying ontologies, learning, and search in isolation, the thesis binds them into a coherent system. The semantic ontology provides the search space and ensures policy compliance. The learned costs improve planning speed and realism. Incremental algorithms make the system responsive to dynamic changes. The result is a neuro-symbolic evacuation planner that is formal, adaptive, and practical.

2.8 Summary

This chapter has surveyed the landscape of indoor emergency evacuation planning, knowledge representation, search algorithms, and machine learning. The review identified significant gaps at the intersection of these domains: existing work treats semantic constraint encoding, dynamic re-planning, learned cost estimation, and neuro-symbolic integration as separate problems. No prior work combines all four within a unified framework.

The thesis fills this gap by developing an integrated neuro-symbolic framework grounded in an OWL–SWRL ontology, powered by incremental heuristic search, and enhanced with BiLSTM-learned transition costs. The chapters that follow demonstrate how each component contributes to a practical evacuation planning system. The heuristic search formulation preserves admissibility under the assumptions of the model; admissibility is a search-theoretic optimality property and does not by itself constitute a guarantee of real-world occupant safety (Section 6.9.2).

3 Ontological Domain Modelling for Indoor Evacuation

3.1 Introduction

Indoor emergency evacuation presents a planning problem whose complexity derives not from the size of the search space alone, but from the semantic richness of the built environment. Rooms differ in function, access policy, and occupancy; corridors differ in width, fire rating, and connectivity; staircases impose vertical movement constraints; and exits carry building-specific regulations governing who may use them and under what conditions. A planning framework that reduces all of these distinctions to a uniform weighted graph discards precisely the information that determines whether a route is feasible, safe, and policy-compliant.

This chapter presents the ontological domain model that underpins the entire neuro-symbolic evacuation planning framework. The model encodes the built environment and evacuation semantics as a formal knowledge base expressed in the Web Ontology Language (OWL) with transition rules expressed in the Semantic Web Rule Language (SWRL). The knowledge base is structured around four interlinked micro-models:

1. **Topological Model**: the static physical structure of the building (spaces, connectivity, floor membership);
2. **Situations Model**: the semantic states an occupant can occupy during evacuation;
3. **Actions Model**: the admissible transitions between situations, constrained by ontology rules;
4. **Events Model**: the emergency occurrences that alter the state of the environment and trigger re-planning.

Together, these four models define a knowledge base $K = (T, S, A, E)$, where T is the topological model, S the set of situations, A the set of actions, and E the

set of events. The planner operates over the state-transition graph induced by K: situations serve as nodes, actions serve as edges, and events modify edge costs or prune edges at run time.

The chapter is organised as follows. Section 3.2 establishes the design principles and requirements that motivate the ontological approach. Sections 3.3 to 3.6 present each of the four micro-models in turn, defining their OWL class hierarchies, property definitions, and SWRL transition rules. Section 3.7 instantiates the framework on three structurally distinct buildings, a Tower, a Hospital, and a School, demonstrating both the expressiveness of the model and the reusability of the generic rule core. Section 3.8 formalises the TBox/ABox separation that enables architectural transferability. Section 3.9 clarifies the terminological distinction between paths and routes. Section 3.10 summarises the chapter.

3.2 Design Principles and Requirements

The ontological model is designed to satisfy five requirements that jointly distinguish it from the geometric graph representations used in conventional evacuation planners.

The five requirements below are derived directly from the research problem formulated in Section 1.1 and the gap clusters synthesised in Section 2.7. Requirements 1 and 2 respond principally to the semantic constraint-encoding challenge by requiring semantically typed spaces and ontology-sanctioned transitions. Requirement 3 responds to the dynamic re-planning challenge by requiring the representation to accommodate event-driven changes affecting the planning state space. Requirement 4 reflects the need for a reusable semantic representation that is not tied to a single building topology. Requirement 5 connects the representation to the search and learned-cost challenges by requiring sufficient semantic structure for heuristic derivation. These requirements therefore precede and motivate the subsequent architecture rather than being inferred retrospectively from the BiLSTM implementation.

Requirement 1: Semantic typing of spaces. Each node in the planning graph must carry a semantic type (room, corridor, vestibule, staircase, exit) drawn from a formal class hierarchy. This typing enables the planner to distinguish between spaces that are geometrically similar but semantically different, for example, a hospital ward and an operating theatre occupy comparable floor areas but impose different access policies during evacuation.

Requirement 2: Policy-aware transitions. Edges in the planning graph must correspond to ontology-sanctioned transitions. A transition from one situation to another is admissible only if it satisfies all applicable prescriptive rules, descriptive rules, and constraints encoded in SWRL. This ensures that every route the planner produces is not merely geometrically shortest but also policy-compliant.

Requirement 3: Event-driven dynamism. The model must support run-time modification of the state space. When an emergency event occurs (fire, blocked corridor, structural failure), the event is asserted into the ontology, the SWRL reasoner refreshes the set of admissible transitions, and the planner operates over the updated graph. The ontological representation must therefore separate static topology from dynamic state.

Requirement 4: Reusability across buildings. The transition rules encoding generic evacuation behaviour (exiting rooms, traversing corridors, descending staircases, reaching exits) must be reusable across structurally different buildings without modification. Building-specific rules (floor-specific descent patterns, restricted zones, multi-exit routing) must be isolated in a separate extension layer. This separation enables the framework to be instantiated on a new building by providing a new assertional layer and a new extension rule set, without altering the generic core.

Requirement 5: Heuristic derivability. The class hierarchy and transition structure must provide sufficient information for the planner to derive an admissible heuristic. Specifically, the minimum number of state-class transitions from the current situation class to the goal class, weighted by average action costs, must be computable via breadth-first search over the class-level graph. This semantic heuristic $h_o(n)$ is defined formally in Chapter 4.

These five requirements motivate the choice of OWL 2 as the representation language: OWL provides a formal class hierarchy with inheritance, object and data properties with domain and range restrictions, and individual assertions that populate the class hierarchy for a specific building. SWRL extends OWL with Horn-clause rules that encode the transition logic.

3.3 Building Topology Ontology

The Topological Model is the static component of the knowledge base. It encodes the physical structure of the building: the types of spaces, their connectivity, and

their assignment to floors.

3.3.1 OWL Class Hierarchy

The building topology is represented as an OWL 2 class hierarchy rooted at the concept `SpatialEntity`. The principal subclasses are:

- `Room`: an enclosed occupiable space. Subclasses include `InnerRoom` (a room accessible only through another room), `Office`, `Lab`, `LectureHall`, and building-specific types such as `Ward`, `ICU`, `OperatingTheatre` (Hospital) and `Workshop`, `ArtRoom`, `MusicRoom` (School).
- `Corridor`: a linear circulation space connecting rooms, vestibules, and other corridors. Subclasses include `LinkCorridor` (a corridor connecting two blocks in the Hospital) and `JunctionCorridor` (a fire-rated corridor connecting a teaching wing to the central hub in the School).
- `Vestibule`: a transitional space between a corridor and a staircase or exit. Vestibules serve as decision points where the planner evaluates whether to continue horizontally or transition vertically.
- `Staircase`: a vertical circulation element connecting floors. Each staircase instance is associated with a pair of floors via the `connectsFloor` property.
- `Floor`: a horizontal subdivision of the building. Floors contain rooms, corridors, and vestibules.
- `Exit`: a point of egress from the building. Exit instances are typed by their function: `MainEntrance`, `FireExit`, `EmergencyDepartmentEntrance`.
- `Building`: the top-level container. Each building instance aggregates floors, staircases, and exits.

The class hierarchy is structured so that generic evacuation rules can be written at the superclass level. For example, the generic rule for exiting a room applies to all subclasses of `Room`, regardless of whether the concrete instance is a `Ward` or a `Workshop`. Building-specific behaviour is captured by subclass-specific rules in the extension layer.

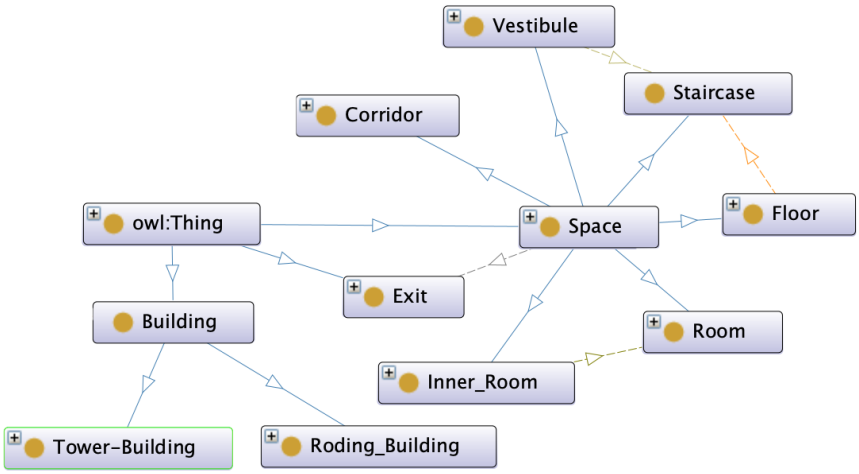

Figure 3.1: OWL class hierarchy for the Building Topology Ontology. Generic classes (grey) are shared across all buildings; building-specific subclasses (coloured) are defined in the extension layer.

3.3.2 Object and Data Properties

The relationships between spatial entities are encoded as OWL object properties. The principal properties are:

- `isAdjacentTo` (symmetric), asserts physical adjacency between two spatial entities. This is the primary connectivity relation.
- `connectsTo`: a directed variant of adjacency used where traversal direction matters (e.g., a one-way fire door).
- `isOnFloor`: assigns a spatial entity to a floor.
- `connectsFloor`: associates a staircase with the pair of floors it connects.
- `hasExit`: associates a building with its exit points.
- `isPartOf`: a mereological relation composing spatial entities into larger units (rooms into floors, floors into buildings).

Data properties capture quantitative attributes:

- `hasCapacity` (integer), the maximum occupancy of a space.

- `hasTraversalCost` (float), the base cost of traversing a space, reflecting physical distance, width, and expected congestion.
- `hasFloorLevel` (integer), the ordinal floor number, used by the semantic heuristic to compute vertical cost.
- `isRestricted` (boolean), flags spaces with restricted access policies (e.g., ICU, operating theatre).

Domain and range restrictions are enforced in OWL. For example, the domain of `isOnFloor` is restricted to the union of `Room`, `Corridor`, and `Vestibule`, while its range is restricted to `Floor`. These restrictions prevent ontological inconsistencies such as asserting that a staircase is on a floor (staircases connect floors but do not belong to a single floor).

3.3.3 Topological Graph Representation

The ontology induces a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that the planner uses for search. Each spatial entity instance in the ABox corresponds to a node $v \in \mathcal{V}$, and each `isAdjacentTo` or `connectsTo` assertion corresponds to an edge $e \in \mathcal{E}$. Edge costs are initialised from the `hasTraversalCost` data property and may be modified at run time when events alter the state of the environment.

Definition 3.1 (Topological Graph)**.** Given a building ontology $\mathcal{O}$ with spatial entity instances $\{v_1, \ldots, v_n\}$ and adjacency assertions $\{(v_i, v_j) \mid \texttt{isAdjacentTo}(v_i, v_j) \in \mathcal{O}\}$, the topological graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{E} = \{(v_i, v_j) \mid \texttt{isAdjacentTo}(v_i, v_j) \in \mathcal{O}\}$. Each edge carries a cost $c(v_i, v_j) = \texttt{hasTraversalCost}(v_j)$, representing the cost of entering node v_j from v_i.

The `QueryNeighbors` function (Equation 3 in the manuscript) retrieves the set of adjacent nodes for a given spatial entity by querying the ontology:

$$\text{neighbors}(v) = \texttt{QueryNeighbors}(v, \mathcal{O}). \tag{3.1}$$

This function returns both explicitly asserted adjacency relations and inferred connections derived by the SWRL reasoner, ensuring that the planner has access to the full connectivity structure of the building.

3.4 Situation Ontology

The Situations Model captures the semantic states an occupant can occupy during evacuation. Each situation is a unary predicate describing the occupant's current context: their location, the structural properties of the space they occupy, and any safety or accessibility constraints that apply.

3.4.1 State-Class Definitions

Situations are defined as OWL classes, each representing a category of occupant state. The principal situation classes form an ordered sequence reflecting the typical evacuation progression from inside the building to outside:

- S_0: `Person_In_Inner_Room`, the occupant is in an inner room (a room accessible only through another room). This is the most deeply nested starting state.
- S_1: `Person_In_Room`, the occupant is in a standard room (office, lab, lecture hall, ward, classroom).
- S_2: `Person_In_Corridor`, the occupant has exited a room and is in a corridor.
- S_3: `Person_In_Vestibule`, the occupant is in a vestibule, a transitional space between a corridor and a staircase or exit.
- S_4: `Person_On_Staircase`, the occupant is on a staircase, transitioning between floors.
- S_5: `Person_On_Floor`, the occupant has arrived on a new floor and must navigate horizontally to the next staircase or exit.
- S_6: `Person_On_Ground_Floor`, the occupant is on the ground floor and can proceed to an exit.
- S_7: `Person_Outside_Building`, the occupant has exited the building. This is the goal state.

Building-specific situation classes extend this generic set. For the Hospital, additional situations include `Person_In_Ward`, `Person_In_ICU`, `Person_In_Operating_Theatre`,

and `Person_In_Link_Corridor`. For the School, additional situations include `Person_In_Workshop`, `Person_In_Main_Hall`, `Person_In_Dining_Hall`, and `Person_In_Junc` These building-specific situations are modelled as subclasses of the generic situations, preserving the class hierarchy and enabling the generic evacuation rules to apply.

Each situation instance encapsulates three dimensions of context:

1. **Location**: the spatial entity the occupant currently occupies.
2. **Structural properties**: the type and floor of the current space, determining which transitions are physically possible.
3. **Safety and accessibility**: whether the current space is restricted, hazardous, or has exceeded its capacity.

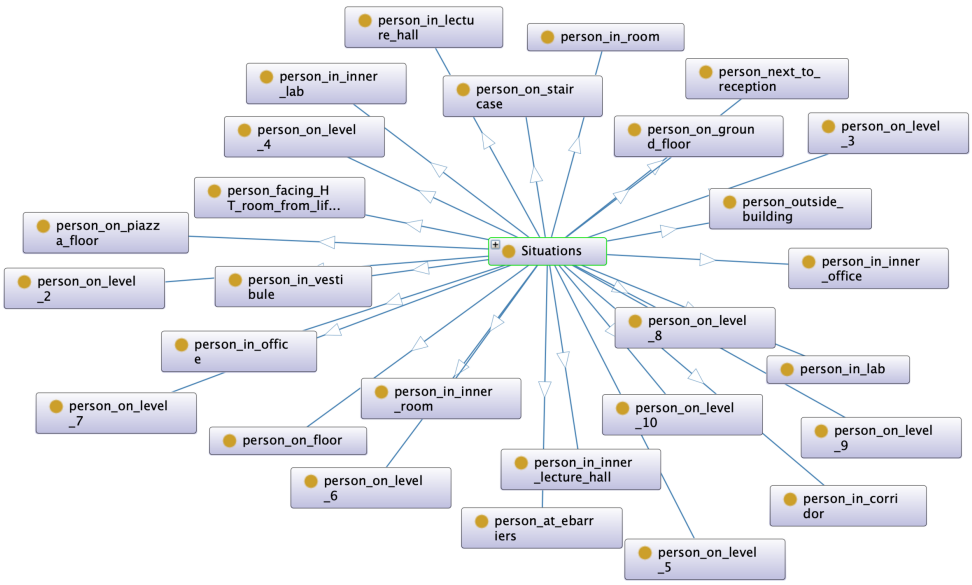

Figure 3.2: Taxonomy of situations in the evacuation ontology. Generic situation classes (S_0–S_7) are shown in grey; building-specific extensions in colour.

3.4.2 State-Transition Graph

The situations and actions together induce a state-transition graph that the planner traverses during search. Each situation serves as a node, and each action defines a directed edge from a source situation to a target situation.

Definition 3.2 (State-Transition System)**.** A state-transition system is a tuple $\Sigma = (S, A, \gamma, s_0, S_g)$ where:

- S is a finite set of situations (states);
- A is a finite set of actions;
- $\gamma : S \times A \to S$ is a partial transition function, where $\gamma(s, a) = s'$ if and only if action a is admissible in situation s and results in situation s';
- $s_0 \in S$ is the initial situation;
- $S_g \subseteq S$ is the set of goal situations (all instances of `Person_Outside_Building`).

The transition function γ is partial because not every action is admissible in every situation. Admissibility is determined by the SWRL rules: an action a is admissible in situation s if and only if the antecedent of the corresponding SWRL rule is satisfied by the current ontology state. Formally, given the current event state e_t, a transition is admissible if:

$$(s_t, a_t) \to s_{t+1} \quad \text{if and only if } \mathcal{R}_{\text{pres}}(s_t) = \text{true}, \tag{3.2}$$

where $\mathcal{R}_{\text{pres}}(s_t)$ denotes the conjunction of all prescriptive rules applicable in situation s_t.

Each action induces a cost:

$$c = \texttt{Cost}(\text{current}, \text{neighbor}), \tag{3.3}$$

which may reflect physical distance, vertical movement penalties, or safety-related preferences. The cost function is used by the planner's evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the accumulated cost from the initial situation to node n and $h(n)$ is the heuristic estimate of the remaining cost to the goal.

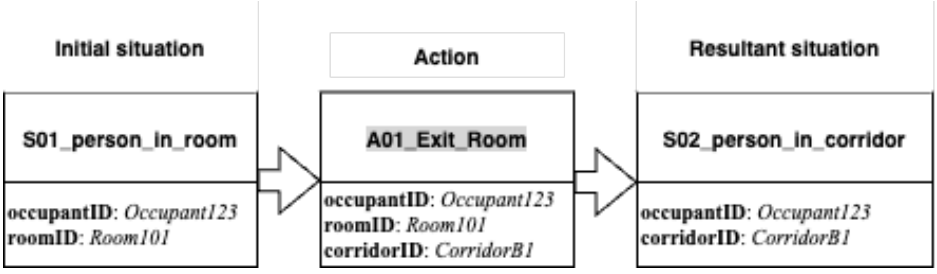

Figure 3.3: Situation parameter transitions. Each transition from an initial situation through an action to a resultant situation is parameterised by the occupant, location, and space properties.

3.5 Event Ontology and Taxonomy

The Events Model captures emergency occurrences that alter the state of the environment during evacuation. Events are the mechanism by which the ontology transitions from a static representation of the building to a dynamic model of evolving hazards.

3.5.1 Impact and Non-Impact Events

Events are classified into two principal categories based on their effect on occupant safety and path viability:

Impact Events directly alter the safety or traversability of the building. An Impact Event either blocks a previously traversable path, degrades the safety of a space, or introduces a new hazard. The three Impact Event types parameterised in the Tower Building evaluation are:

- E_{01}: `Fire_Alarm_Triggered`, activates evacuation mode. All occupants must proceed to the nearest exit. This event does not block any path but changes the planning objective from normal circulation to evacuation.
- E_{02}: `Obstacle_Blocking_Vestibule_Door`, a physical obstruction (collapsed ceiling, fallen debris) renders a vestibule impassable. The corresponding edge in the topological graph is removed, potentially isolating upper floors from the staircase.
- E_{03}: `Bottleneck_Impeding_Egress`, congestion or structural narrowing in a corridor or staircase reduces traversal capacity. The edge cost is increased rather than the edge being removed, reflecting degraded but not eliminated access.

Non-Impact Events are informational. They do not immediately threaten occupant safety or block paths, but they may influence planning decisions by providing contextual information. Examples include smoke detection in a distant zone (which may indicate future path degradation) or an all-clear signal for a previously hazardous area.

Each event instance carries three parameters:

- **Classifier**: the event type within the taxonomy (Impact or Non-Impact, and the specific subtype).
- **Temporal parameter**: the time at which the event occurs, relative to the evacuation start.
- **Description**: the spatial entities affected by the event (which corridor is blocked, which vestibule is obstructed).

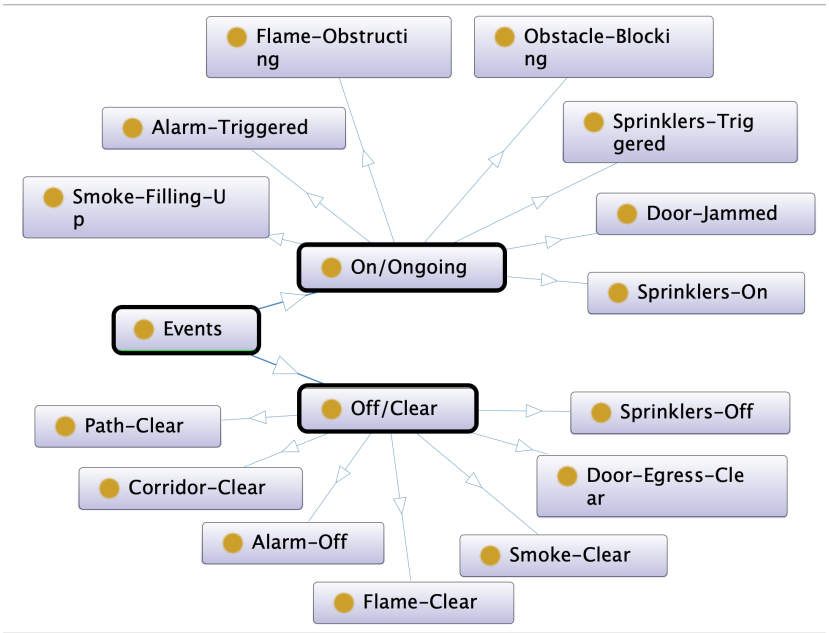

Figure 3.4: Event taxonomy. Events are classified as Impact (directly affecting path viability or safety) or Non-Impact (informational). The three parameterised Impact Event types (E_{01}–E_{03}) used in the Tower Building evaluation are highlighted.

3.5.2 Event–Situation Interaction

When an event occurs, it is asserted into the ontology as a typed fact. The SWRL reasoner then refreshes the set of admissible transitions by re-evaluating the rule antecedents against the updated ontology state. This event-driven update mechanism has three consequences for the planner:

1. **Edge removal.** If an event renders a space impassable (e.g., E_{02} blocking a vestibule), all edges incident on the corresponding node are removed from the topological graph. The planner must find an alternative route.

2. **Edge cost modification.** If an event degrades but does not eliminate access (e.g., E_{03} causing a bottleneck), the edge cost is increased. The planner may still use the affected path but will prefer alternatives if they are cheaper.

3. **New situation assertion.** An event may create a new situation that did not previously exist. For example, blocking the only staircase on an upper floor creates the situation `Person_Stuck_At_Barriers`, which triggers the backtracking mechanism described in Chapter 5.

The framework models hazard events as discrete ontology updates: a fire alarm activates evacuation mode, a structural incident blocks a corridor segment, and a bottleneck renders a vestibule impassable. Each such event is asserted into the ontology as a typed fact that alters the admissible transition set. This representation is well suited to events that change the binary availability or traversal cost of building segments. Phenomena that are inherently continuous or spatially distributed, such as progressive smoke propagation, crowd density waves, panic-driven behavioural shifts, and heterogeneous occupant mobility, are not modelled by the current ontology and SWRL layer. The BiLSTM learns from symbolic state–action trajectories generated under these discrete events, not from continuous physical simulations or raw sensor streams. Consequently, the framework's adaptive capabilities operate over discrete, event-driven hazard changes within a semantically typed state space, rather than over the full spectrum of physical evacuation dynamics.

3.6 Action Ontology and SWRL Rules

The Actions Model defines the admissible movements between situations. Each action is a binary predicate relating a source situation to a target situation, encoded as an OWL object property and constrained by SWRL rules.

3.6.1 Navigation Actions

Actions are classified into three categories based on their role in the evacuation process:

Navigational Actions prescribe physical movement through the building. These include:

- `exit_inner_room`(x, y), move from an inner room to the containing room.
- `exit_room`(x, y), move from a room to the adjacent corridor.
- `enter_vestibule`(x, y), move from a corridor to a vestibule.
- `exit_to_staircase`(x, y), move from a vestibule to a staircase.
- `head_downstairs_to_floor`(x, y), descend one floor via a staircase.
- `exit_building_main_door`(x, y), exit the building through the main entrance.

These actions encode both horizontal navigation (room → corridor → vestibule) and vertical navigation (staircase → lower floor). The full traversal chain from innermost room to building exterior follows the sequence: `inner_room` → `room` → `corridor` → `vestibule` → `staircase` → `floor` → `ground_floor` → `outside`.

Corrective Actions respond to events that invalidate the current plan. These include rerouting actions that redirect an occupant to an alternative corridor or exit when the planned route is blocked.

Backtracking Actions are a special category of corrective action that reverse the occupant along previously traversed path segments. Unlike standard corrective actions (which redirect to a new forward path), backtracking actions return the occupant to an earlier decision point from which a different route can be attempted. The BiLSTM backtracking prediction head described in Chapter 5 provides predictive guidance for when to invoke these actions.

Each action a is encoded as an SWRL rule of the form:

$$\text{Situation}_{\text{source}}(?x) \wedge \text{Conditions}(?x, ?y) \Rightarrow \text{Situation}_{\text{target}}(?x, ?y)$$

where the antecedent specifies the source situation and any additional conditions (adjacency, floor membership, absence of hazards), and the consequent asserts the target situation.

3.6.2 SWRL Rule Categories: Prescriptive, Descriptive, and Policy

The SWRL rules that govern the action space are organised into four categories, each serving a distinct role in the planning process.

Prescriptive Rules define mandatory evacuation actions that must be taken under specific conditions. They constrain the action space by allowing only transitions that satisfy predefined safety conditions:

$$(s_t, a_t) \rightarrow s_{t+1} \quad \text{if and only if } \mathcal{R}_{\text{pres}}(s_t) = \text{true}. \tag{3.4}$$

For example, a prescriptive rule may enforce that once a fire alarm is triggered, all occupants must proceed toward exits rather than returning to rooms. These rules ensure compliance with evacuation policies and safety regulations.

Descriptive Rules capture factual relationships within the environment that are not explicitly encoded in the topological graph. They enable the ontology reasoner to infer implicit connections, such as adjacency between spaces on different floors via a staircase, or accessibility relationships derived from building design standards. These inferred relations are queried during planning through the function:

$$\texttt{QueryInferredConnections}(\text{current}, \mathcal{O}).$$

Policy Rules express evacuation preferences rather than strict constraints. They influence the planning process by affecting heuristic evaluation and cost assignment rather than by enforcing hard exclusions. Examples include prioritising safer routes, avoiding congested areas, or favouring ramps over stairs for mobility-impaired evacuees.

Constraint Encoding represents physical and operational limitations such as blocked passages, closed doors, or capacity restrictions. Constraints dynamically prune the state-transition graph by removing infeasible neighbours during node expansion.

Together, these four rule categories ensure that all paths explored by the planner are semantically valid, policy-compliant, and context-aware while remaining flexible to dynamic environmental changes.

The SWRL rules are organised into two layers: a *generic core* of 47 rules encoding universal evacuation transition behaviours, and a *building-specific extension layer* encoding navigation actions unique to a particular building topology. This layered architecture is central to the framework's reusability and is discussed further in Section 3.8.

3.7 Case Studies: Tower, Hospital, and School Buildings

The ontological framework is instantiated on three structurally distinct buildings, each exercising different aspects of the knowledge representation and planning components. The three-building set is selected to ensure that positive evaluation results cannot be attributed to overfitting to a single building geometry.

3.7.1 Tower Building

3.7.1.1 Building Description and Topology Instantiation

The Tower Building is a 10-storey academic building at London Metropolitan University. It features a single vertical core with a rectangular footprint, one central staircase, and one ground-floor exit (the main entrance). The building's evacuation challenge is dominated by vertical descent through a single staircase bottleneck.

The ontology instantiation of the Tower Building produces a state-transition graph with 26 nodes and approximately 100 edges. Each floor contributes nodes for rooms, corridors, and vestibules; the single staircase provides vertical connections between adjacent floors. The linear topology, where all occupants must converge on a single descent path, makes the Tower Building particularly sensitive to mid-path hazard events that block the staircase or vestibule, creating the conditions under which backtracking is required.

The Tower Building serves as the primary evaluation environment. It provides the basis for all empirical results reported in Chapter 6: the 50 dynamic scenarios spanning three complexity tiers (Simple, Semi-Complex, Complex) are parameterised on the Tower Building topology, and the BiLSTM is trained on trajectories generated from Tower Building evacuations.

3.7.1.2 Ontology Population and Verification

The Tower Building ontology is embedded in the main BEM.owl file. It contains approximately 80 spatial entity individuals, 50 occupant individuals, and 120 connectivity assertions. The building-specific extension layer comprises 56 SWRL rules

encoding floor-specific descent transitions (e.g., `head_downstairs_to_floor_2`, `head_downstairs_to_ground_floor`), room-specific exit actions for different room types on each floor, and vestibule navigation actions.

Ontology verification was performed in two stages. First, the OWL reasoner (Pellet) was used to check the ontology for logical consistency, no unsatisfiable classes, no contradictory assertions, and no violated domain/range restrictions. Second, the SWRL rules were tested by asserting sample occupant positions and verifying that the reasoner correctly inferred the set of admissible transitions. For example, asserting `Person_In_Room(occupant1, Room_F5_01)` and verifying that the reasoner infers `exit_room(occupant1, Corridor_F5)` as an admissible action, but does not infer any action that would bypass the corridor and proceed directly to the staircase.

3.7.2 Hospital Building

3.7.2.1 Building Description and Topology Instantiation

The Hospital Building is modelled as a generic NHS-style district general hospital, a 3-storey dual-block structure (Block A and Block B) connected at each floor level by link corridors. The design is grounded in NHS Health Building Notes HBN 00-01 and HBN 00-04, which specify spatial requirements for clinical environments.

The Hospital's evacuation challenge differs fundamentally from the Tower's. Rather than a single vertical bottleneck, the Hospital presents a branching decision problem: occupants must choose between blocks (if they can reach a link corridor) and between three ground-floor exits (main entrance, emergency department entrance, and rear fire exit). Additionally, the ICU and operating theatre are modelled as *restricted zones*, spaces with one-way evacuation semantics where occupants can exit but cannot re-enter during evacuation. This restriction tests the framework's ability to enforce prescriptive rules that constrain the action space based on space type and access policy.

The Hospital ontology (Hospital.owl) contains approximately 65 spatial entity individuals, 40 occupant individuals, and 100 connectivity assertions. It has two staircases (one per block) and three exits.

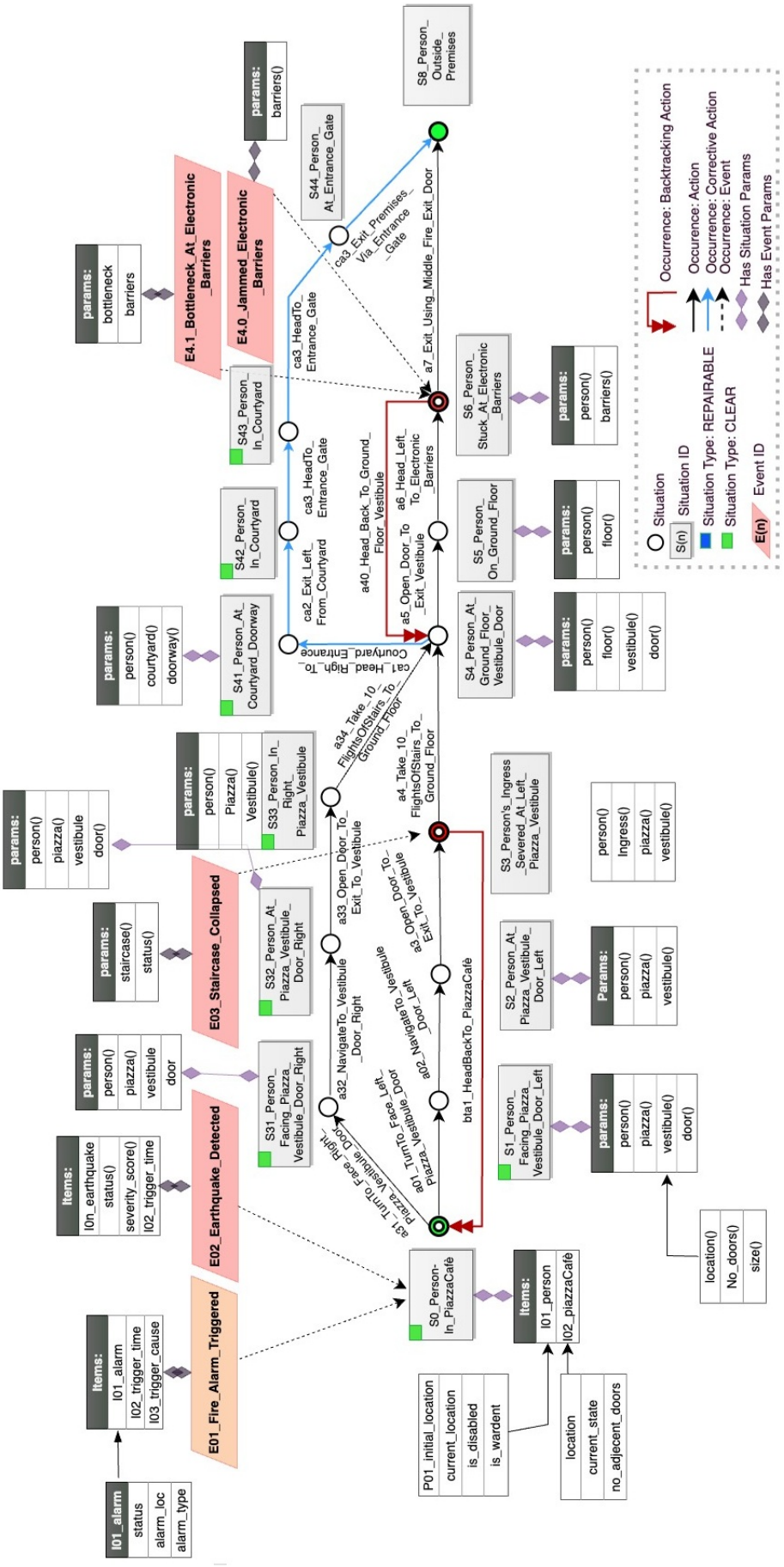

Figure 3.5: Ontology-based scenario representation for the Tower Building over multiple floors. The figure illustrates the spatial entities, connectivity relations, and hazard events that parameterise Complex evacuation scenarios requiring re-planning and backtracking.

3.7.2.2 Ontology Population and Verification

The Hospital building-specific extension layer comprises 38 SWRL rules. Elevator-related rules are excluded, as elevators are not admissible evacuation routes under standard fire safety protocols. These rules encode hospital-specific navigation actions including `exit_ward`, `exit_icu`, `cross_link_corridor`, and `exit_to_ed_entrance`, as well as restricted-zone constraints that prevent re-entry into the ICU and operating theatre once an occupant has exited.

The Hospital ontology reuses the 47-rule generic SWRL core unchanged. This demonstrates that the generic evacuation traversal chain (room → corridor → vestibule → staircase → floor → exit) applies to the Hospital topology without modification, despite the Hospital's dual-block layout, restricted zones, and multiple exits. The building-specific extension rules handle only the Hospital-unique navigation patterns that cannot be expressed at the generic level.

Verification followed the same two-stage process as the Tower Building: OWL consistency checking via the Pellet reasoner, followed by SWRL rule testing with sample occupant positions. Additional verification was performed on the restricted-zone rules to confirm that the reasoner correctly prevents re-entry transitions.

3.7.3 School Building

3.7.3.1 Building Description and Topology Instantiation

The School Building is modelled as a generic DfE-style finger-block secondary school, a 2-storey structure with three teaching wings (Wing A, Wing B, Wing C) radiating from a central hub. The design is grounded in DfE Building Bulletin BB100 and BB103, which specify spatial requirements for educational environments. The hub contains high-occupancy spaces (main hall, dining hall, library) and connects to the main entrance. Each wing terminates in a fire exit, giving the building four exits in total.

The School's evacuation challenge stresses the multi-exit heuristic comparison component of the framework. Occupants in wings must decide whether to proceed to the wing-end fire exit or traverse junction corridors to the hub for the main entrance. Junction corridors are fire-rated corridors with fire doors on both sides,

creating natural compartmentation boundaries. If a junction corridor is blocked, the wing is isolated and occupants must use the wing-end fire exit exclusively.

The School ontology (School.owl) contains approximately 90 spatial entity individuals, 50 occupant individuals, and 130 connectivity assertions. It has three staircases (one per wing, at wing ends), four exits, and produces a state-transition graph with approximately 32 nodes, the largest and most highly branching of the three buildings.

3.7.3.2 Ontology Population and Verification

The School building-specific extension layer comprises 39 SWRL rules encoding school-specific navigation actions including `exit_main_hall`, `exit_dining_hall`, `exit_library`, `enter_junction_corridor`, `exit_to_hub`, and `exit_to_wing`. Unlike the Hospital, the School has no restricted zones; its complexity derives from the high branching factor at the hub and the multiple independent egress paths.

As with the Hospital, the School ontology reuses the 47-rule generic SWRL core unchanged. The extension rules include `go_up` actions to support backtracking scenarios where an occupant on the ground floor must return to the upper floor to access an alternative wing.

3.8 TBox/ABox Separation and Architectural Transferability

The ontological framework is designed with a clear separation between its terminological layer (TBox) and its assertional layer (ABox). This separation is the mechanism by which the framework achieves reusability across structurally distinct buildings.

The **TBox** defines the generic class hierarchy for spatial entities, person states, actions, events, and object properties. It contains no building-specific individuals or assertions. All three buildings, Tower, Hospital, and School, share the same TBox, defined in BEM.owl. The generic SWRL core (47 rules) operates exclusively at the TBox level, using generic class names (`Room`, `Corridor`, `Vestibule`, `Staircase`, `Floor`, `Exit`) rather than building-specific individuals.

The **ABox** populates the TBox schema with individuals, connections, and floor

assignments specific to a particular building. Each building has its own ABox:

- Tower Building: ABox embedded in BEM.owl (∼80 spatial individuals, 50 occupants, ∼120 connectivity assertions).
- Hospital Building: ABox in Hospital.owl (∼65 spatial individuals, 40 occupants, ∼100 connectivity assertions).
- School Building: ABox in School.owl (∼90 spatial individuals, 50 occupants, ∼130 connectivity assertions).

The **SWRL transition rules** are likewise organised into two layers:

- **Generic core** (47 rules): encodes universal evacuation behaviours that apply to any building. These rules reference only TBox-level class names and are shared across all three buildings without modification.
- **Building-specific extensions**: encode navigation actions unique to a particular building topology. Tower: 56 rules; Hospital: 38 rules; School: 39 rules. These extensions introduce building-specific situation types, navigation actions, exit chains, and constraints.

Table 3.1: Summary of the SWRL rule architecture across three buildings. The generic core is shared unchanged; only the extension layer varies.

Property	**Tower**	**Hospital**	**School**
Generic core rules	47	47	47
Extension rules	56	38	39
Total SWRL rules	103	85	86
Unique situation types	Floor-specific	Ward, ICU, OT, ED	Workshop, Hall, Library
Exits	1	3	4
Staircases	1	2	3
Restricted zones	None	ICU, OT	None

To adapt the framework to a new building, three steps are required: (1) define a new ABox populating the TBox classes with the new building's spatial entities and connectivity; (2) write a building-specific SWRL extension layer encoding the navigation actions unique to the new topology; (3) retain the generic TBox and 47-rule generic core without modification. This process has been completed for all three buildings, demonstrating what is termed *architectural transferability*: the ability of the framework's knowledge representation layer to be instantiated for a structurally distinct building without modifying the generic rules.

Architectural transferability is distinguished from full empirical cross-building validation, which additionally requires running the planner on the new building, measuring performance metrics (T_{avg}, S_r, N_{nodes}), and retraining the BiLSTM on the new building's trajectories. The current evaluation provides full empirical results for the Tower Building; the Hospital and School ontologies demonstrate architectural transferability of the knowledge layer.

3.9 Path vs. Route: A Terminological Distinction

The thesis maintains a consistent terminological distinction between *paths* and *routes*, reflecting a design decision embedded in the ontological framework.

A **path** is a static, physical sequence of connected spaces in the building topology. A path has four key properties: (a) it is related to physical location; (b) it is fixed and non-dynamic; (c) it is independent from individuals; and (d) it ensures spatial continuity, each consecutive pair of spaces in the sequence must be adjacent in the topological graph.

A **route** is a dynamic, event-responsive navigation plan composed of one or more paths. A route has four contrasting properties: (a) it is dynamic and subject to re-routing when events alter the environment; (b) it may be composed of several paths, switching between them at decision points; (c) it is generated over existing paths and may span multiple paths when the planner replans; and (d) it is not directly obstructable, paths may be blocked, making a route over them infeasible, but the route itself is a planning construct rather than a physical entity.

This distinction has practical consequences for the planner. A single evacuation may involve multiple paths if events force re-planning. The planner computes a route as a sequence of actions, each of which traverses a segment of a physical path. When an event invalidates the current path segment, the planner computes a new route that may follow an entirely different set of paths. The BiLSTM-guided planner further refines this by predicting which paths are likely to become infeasible, enabling preemptive route changes before a dead-end is reached.

3.10 Summary

This chapter has presented the ontological domain model that forms the foundation of the neuro-symbolic evacuation planning framework. The key contributions of this chapter are:

1. **Four-model knowledge base.** The built environment and evacuation semantics are encoded as a formal knowledge base $K = (T, S, A, E)$ comprising four interlinked OWL/SWRL ontologies: Topology, Situations, Actions, and Events. This representation provides semantic typing, policy-aware transitions, and event-driven dynamism.

2. **State-transition formalism.** A state-transition system $\Sigma = (S, A, \gamma, s_0, S_g)$ is defined over the ontological models, providing the formal basis for heuristic search. Situations serve as nodes, actions as edges, and events modify the graph at run time.

3. **Layered SWRL rule architecture.** Transition rules are organised into a reusable generic core (47 rules) and building-specific extension layers (38–56 rules per building). This separation enables architectural transferability without modifying the generic rules.

4. **Three-building instantiation.** The framework has been instantiated on three structurally distinct buildings, a Tower (10-storey, single staircase, 1 exit), a Hospital (3-storey dual-block, restricted zones, 3 exits), and a School (2-storey finger-block, hub routing, 4 exits), demonstrating the expressiveness and reusability of the ontological approach.

5. **Path/Route distinction.** The terminological and ontological separation between static paths and dynamic routes provides a precise vocabulary for describing the planner's behaviour under event-driven re-planning.

The ontological models and formalisms established in this chapter are used by the planning algorithms presented in the following chapters: the semantic heuristic $h_o(n)$ and LPA* search in Chapter 4, and the BiLSTM-guided hybrid heuristic $h'(n)$ in Chapter 5.

4 Semantic Heuristic Search and Event-Driven Re-planning

4.1 Introduction

The preceding chapter established a formal knowledge representation framework for emergency evacuation domains, grounded in description logic and SWRL rule inference over ontology-derived situation instances. This chapter turns to the problem of computing optimal or near-optimal evacuation routes from that semantic representation. The core challenge is to bridge the gap between high-level logical descriptions of situations,such as *person in room A, room A connected to corridor B, corridor B partially obstructed*,and efficient graph-based path planning algorithms.

This chapter develops a complete pipeline for semantic heuristic search and reactive re-planning:

1. We adapt the Lifelong Planning A* (LPA*) algorithm to operate over ontology-derived state-transition graphs, in which nodes represent situation instances inferred from the knowledge base $K = (T, S, A, E)$ and edges represent SWRL rule instantiations.

2. We propose a semantic heuristic function $h_o(n)$ that exploits the structure of situation type hierarchies to provide an admissible, computationally efficient estimate of the cost to person evacuation from any situation instance.

3. We implement a static path planning algorithm that uses $h_o(n)$ to compute optimal evacuation routes prior to execution, establishing a baseline against which dynamic re-planning can be measured.

4. We present an event-driven re-planning architecture that monitors evacuation progress at runtime, detects deviations between predicted and actual situation evolution, and triggers incremental graph updates and re-planning to recover from path invalidation, congestion, and other disruptions.

5. We introduce a reactive backtracking mechanism that allows the evacuation plan to retreat to the last feasible decision point when an action becomes infeasible, rather than abandoning the entire plan.

6. We define a taxonomy of evacuation scenarios spanning simple (single event, no invalidation), semi-complex (event plus 1–2 local path obstructions), and complex (multiple obstructions requiring backtracking) cases, and characterize the computational profile of the planning and re-planning algorithms across these tiers.

Throughout, we emphasize that this chapter covers the *purely symbolic* search layer. The extension to hybrid, learning-informed heuristic guidance via BiLSTM is deferred to Chapter 5. The algorithms and scenario evaluation presented here serve as the deterministic baseline against which learned guidance is compared.

4.2 LPA* for Ontology-Derived Graphs

4.2.1 Standard LPA*

Lifelong Planning A* (LPA*) [52] is an incremental variant of A* that excels in dynamic environments where edge weights may change during or after an initial search. Rather than recomputing a path from scratch after a cost update, LPA*reuses previously computed values and only re-expands affected nodes, often recovering a new path far more rapidly than running A* de novo.

The algorithm maintains two key pieces of state for each node n:

- $g(n)$: the cost of the shortest path from the start node s to n found so far.
- $\mathrm{rhs}(n)$: the right-hand side value, a one-step lookahead estimate defined as

$$\mathrm{rhs}(n) = \begin{cases} 0 & \text{if } n = s \\ \min_{n' \in \mathrm{pred}(n)} \Big\{ g(n') + w(n', n) \Big\} & \text{otherwise} \end{cases} \tag{4.1}$$

  where $\mathrm{pred}(n)$ is the set of predecessors of n and $w(n', n)$ is the edge weight.

A node is *locally consistent* when $g(n) = \mathrm{rhs}(n)$. The algorithm maintains a priority queue of inconsistent nodes. Each node n is assigned a priority key

$\mathbf{k}(n) = [k_1(n), k_2(n)]$ where

$$k_1(n) = \min\Big(g(n), \text{rhs}(n)\Big) + h(n), \quad k_2(n) = \min\Big(g(n), \text{rhs}(n)\Big) \tag{4.2}$$

The algorithm expands nodes from the queue in order of increasing key, updates g values, and maintains consistency. When an edge weight changes, the affected nodes are inserted into the queue, their rhs values are recomputed, and expansion resumes. This incremental strategy can yield substantial speedups when the number of affected nodes is small relative to the total graph size.

4.2.2 Adaptation to Semantic State-Transition Graphs

The ontology-guided evacuation planning domain presents a natural fit for LPA*. Rather than a fixed, pre-enumerated graph, the search space consists of situation instances $s_i \in S$ (from the ABox) whose successors are determined dynamically by instantiating SWRL rules. Edge weights are assigned based on semantic rule properties and dynamically updated as physical conditions (congestion, blockage) change.

In this adaptation:

1. **Nodes** correspond to situation instances. A situation instance encodes the current state of the evacuation (e.g., person location, environment status, time). The search begins from the initial situation s_0 (e.g., all occupants in starting rooms, all exits clear) and aims to reach a goal situation in which all persons are outside the building.

2. **Edges** are derived from SWRL rule instantiations. When a rule r has a body that matches situation s_i, the rule's consequent defines a target situation s_j reachable from s_i. The edge (s_i, s_j) is added to the graph dynamically as rules fire.

3. **Edge weights** are assigned per rule type and updated dynamically. Base weights reflect the semantic cost of transitions (e.g., exiting a room = 1.0, traversing a corridor = 1.0, descending a staircase = 2.0). When an event indicates obstruction or congestion, the affected edge weights are modified: a blocked path becomes infinite, a congested zone receives a multiplier (e.g., 3× base cost).

4. **Dynamic updates** occur when environment events are detected. The planning system checks whether previously valid paths remain feasible and recomputes weights. LPA*'s incremental machinery ensures only affected nodes are re-expanded.

The key advantage is that the planner never needs to construct the full graph in advance. Instead, nodes and edges are instantiated on-demand during search, and the semantic rules serve as the implicit graph definition. This allows the planner to scale to large, heterogeneous evacuation domains where the explicit enumeration of all possible situations would be infeasible.

4.3 Semantic Heuristic $h_o(n)$

A good heuristic function is critical to the performance of A* and LPA*. In traditional grid-based or graph-based path planning, heuristics such as Euclidean distance or Manhattan distance are straightforward to compute. In an ontology-derived situation space, however, these geometric heuristics are not applicable. Instead, we propose a *semantic heuristic* $h_o(n)$ that estimates the cost to reach goal from a situation instance by analyzing the structure of situation type hierarchies and SWRL rule chains.

4.3.1 Definition: BFS over State Classes

The heuristic $h_o(n)$ is defined in terms of rule chains at the type level. Given a situation instance s of type σ (e.g., *PersonInRoom*), we ask: what is the minimum number and cost of SWRL rule applications required to transition from a situation of type σ to the goal type *PersonOutsideBuilding*?

The algorithm proceeds as follows:

1. **Type-level BFS**: Perform a breadth-first search over the situation type lattice, starting from the type σ of the given situation instance s. At each step, consider all SWRL rules whose body can fire on a situation of the current type; the consequent of such a rule introduces one or more successor types.

2. **Minimum chain**: Record the shortest path (in terms of rule applications) from σ to the goal type. This path is a *type-level chain*, not an instance-level

trajectory. Let $d(s)$ denote the length of the shortest chain.

3. **Cost aggregation**: For each rule type appearing in the shortest chain, aggregate the base costs. Specifically, if the chain involves $d(s)$ rule applications, and rule type i (with $i = 1, \ldots, d(s)$) has average cost $c_{\text{avg}}(i)$ (computed over all instantiations of that rule type during training or offline analysis), then:

$$h_o(s) = \sum_{i=1}^{d(s)} c_{\text{avg}}(i) \tag{4.3}$$

4. **Caching**: Type-level chains and their costs are precomputed offline and cached. At search time, computing $h_o(n)$ is $O(1)$ or $O(\log k)$ (where k is the number of situation types), making the heuristic extremely efficient.

The semantic structure of the heuristic makes it interpretable: a high h_o value for a situation instance indicates that the instance is far (in terms of logical transitions) from the goal, while a low value suggests the instance is close. This contrasts with black-box heuristics and aligns with domain knowledge encoded in the ontology.

4.3.2 Admissibility Proof

For A* and LPA*to guarantee optimal path discovery, the heuristic must be *admissible*: it must never overestimate the true cost from a node to the goal. We prove that $h_o(n)$ satisfies this condition.

Proposition 4.1 (Admissibility of $h_o(n)$)**.** *Let $h^*(n)$ denote the true minimum cost from situation instance n to the goal. Then $h_o(n) \leq h^*(n)$ for all situation instances n.*

Proof. Consider any situation instance n of type σ. Let $d^*(n)$ denote the true minimum cost (in weighted edge terms) from n to the goal, and let $d_{\text{chain}}(\sigma)$ denote the minimum number of SWRL rules needed to chain from type σ to the goal type.

At minimum, any path from n to the goal must instantiate at least $d_{\text{chain}}(\sigma)$ rules (or follow a longer rule chain with potentially fewer rules but higher per-rule costs). The heuristic $h_o(n)$ uses the minimum rule chain length and sums the average costs for each rule type in that chain.

Now, a key observation: in a dynamic evacuation scenario, edge costs can only *increase* from their base values due to congestion or blockage; costs never decrease

below base. Average costs $c_{\text{avg}}(i)$ used in the heuristic are computed over baseline (non-congested, unobstructed) configurations. Therefore:

$$c_{\text{avg}}(i) \leq c(i)_{\text{actual}} \tag{4.4}$$

where $c(i)_{\text{actual}}$ is the true cost of rule type i in any dynamic state.

Thus:

$$h_o(n) = \sum_{i=1}^{d_{\text{chain}}} c_{\text{avg}}(i) \leq \sum_{i=1}^{d_{\text{chain}}} c(i)_{\text{actual}} \leq d^*(n) = h^*(n) \tag{4.5}$$

The final inequality holds because any optimal path must traverse at least the minimum rule chain, and the actual costs along that path are at least the sum of base costs. □ □

The admissibility of $h_o(n)$ is preserved even in dynamic conditions because the heuristic always underestimates: it uses average baseline costs, and dynamic costs only increase. This guarantees that LPA*with h_o as the heuristic will find optimal evacuation routes.

4.4 Static Path Planning

Before addressing the dynamic, event-driven re-planning problem, we first establish the *static* planning case: computing an evacuation route under the assumption that the environment does not change during execution. This serves as a baseline and validates the heuristic before it is tested under dynamics.

4.4.1 Planning Algorithm

The static planning algorithm is a straightforward application of A* to the ontology-derived state-transition graph:

1. **Initialize**: Set the open list to contain only the initial situation s_0. Set $g(s_0) = 0$ and mark s_0 as the parent of itself.
2. **Main loop**: While the open list is not empty:
   (a) Pop the node n with the lowest $f(n) = g(n) + h_o(n)$ value.

(b) If n is a goal situation (all persons outside building), reconstruct and return the path by backtracking through parent pointers.

(c) Expand n: query the SWRL rule engine to find all situation instances reachable via rule instantiation. For each successor n':

i. Compute the edge weight $w(n, n')$ based on the rule type.

ii. If $g(n) + w(n, n') < g(n')$, update $g(n')$ and insert n' into the open list.

(d) If expansion violates semantic constraints (e.g., a rule body requires a precondition that is not satisfied), mark that successor as inadmissible and do not expand further along that branch.

3. **Return**: If the open list is exhausted, no feasible path exists under the current semantic constraints.

In practice, the rule engine returns neighbors via two queries:

- **QueryNeighbors**(n): returns explicit direct successors of n based on fired SWRL rules.
- **QueryInferredConnections**(n): returns successors inferred via chains of SWRL rules that may not be explicitly modeled but are logically derivable.

The planner uses both queries to ensure no feasible transition is overlooked. The returned paths are guaranteed optimal under the semantic constraints and the admissible heuristic.

4.4.2 Worked Example: Simple Scenario

Consider a simplified two-floor building with three rooms on the ground floor (G1, G2, G3), one staircase (SC), and a vestibule (V) on the first floor leading to the outside. A person begins in room G1 at time $t = 0$.

Situation types relevant to this scenario:

- `PersonInRoom(person, room)`: person is in a room.
- `PersonInCorridor(person, corridor)`: person has moved to a corridor.
- `PersonAtStaircase(person, staircase)`: person is at the foot of a staircase.

- `PersonOnStaircase(person, staircase)`: person is traversing a staircase.
- `PersonInVestibule(person, vestibule)`: person is in the vestibule.
- `PersonOutsideBuilding(person)`: person has exited; goal condition.

SWRL rules encode transitions:

- `R1`: If `PersonInRoom`(p, r) and `room_has_exit`(r, c) and `corridor_clear`(c), then `PersonInCorridor`(p, c). Cost: 1.0 (exiting a room).
- `R2`: If `PersonInCorridor`(p, c) and `corridor_leads_to`(c, sc) and `staircase_accessibl`
then `PersonAtStaircase`(p, sc). Cost: 1.0 (moving through corridor).
- `R3`: If `PersonAtStaircase`(p, sc), then `PersonOnStaircase`(p, sc). Cost: 2.0 (staircase descent).
- `R4`: If `PersonOnStaircase`(p, sc) and `staircase_exit_clear`(sc), then `PersonInVestibule`(p, v). Cost: 1.0 (exiting staircase).
- `R5`: If `PersonInVestibule`(p, v) and `vestibule_exit_clear`(v), then `PersonOutsideBui`
Cost: 1.0 (final exit).

Static planning (no events, no path invalidation):

Starting from $s_0 =$ `PersonInRoom`$(\text{alice}, \text{G1})$, the planner executes A*:

1. $g(s_0) = 0$, $h_o(s_0) = 5.0$ (BFS over types: PersonInRoom $\rightarrow$ PersonInCorridor $\rightarrow$ PersonAtStaircase $\rightarrow$ PersonOnStaircase $\rightarrow$ PersonInVestibule $\rightarrow$ PersonOutsideBuilding, six types, five transitions, total base cost $1.0 + 1.0 + 2.0 + 1.0 + 1.0 = 6.0$; conservative estimate is 5.0). $f(s_0) = 5.0$.
2. Pop s_0. Expand via R1: successor is $s_1 =$ `PersonInCorridor`(alice, c) with $g(s_1) = 0 + 1.0 = 1.0$, $h_o(s_1) = 4.0$ (remaining chain: PersonInCorridor $\rightarrow \ldots \rightarrow$ PersonOutsideBuilding). $f(s_1) = 1.0 + 4.0 = 5.0$.
3. Pop s_1. Expand via R2: successor is $s_2 =$ `PersonAtStaircase`$(\text{alice}, \text{sc})$ with $g(s_2) = 1.0 + 1.0 = 2.0$, $h_o(s_2) = 3.0$. $f(s_2) = 5.0$.
4. Continue similarly through s_3 (PersonOnStaircase), s_4 (PersonInVestibule), to s_5 (PersonOutsideBuilding).
5. Final path: $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$ with total cost $g(s_5) = 0 + 1.0 + 1.0 + 2.0 + 1.0 + 1.0 = 6.0$.

The heuristic guides the planner efficiently toward the goal without exploring irrelevant branches (e.g., person re-entering a room). The admissibility of h_o guarantees that the returned path is optimal.

4.5 Event-Driven Re-planning Architecture

In real emergency evacuation, conditions change. A corridor may become congested, a door may be blocked, or a planned route may become infeasible. The static planning algorithm alone is insufficient. We now introduce an event-driven re-planning architecture that monitors evacuation execution and incrementally recomputes paths when edge cost updates or transition invalidation occur.

4.5.1 Event Classification and Decision Matrix

The system distinguishes events by their impact on the graph structure and whether they trigger re-planning. Table 4.1 summarizes the event types across the three building domains used in evaluation (Tower, Hospital, School).

Table 4.1: Event Types and Re-planning Triggers Across Evaluation Domains

Event ID	**Domain**	**Type**	**Graph Impact**	**Re-plan Trigger**
E01	Tower	Alarm (init.)	Initial situation change	Yes, initialize
E02	Tower	Vestibule obstruction	Edge weight ∞	Yes, LPA*update
E03	Tower	Corridor bottleneck	Edge weight 3$\times$	Yes, LPA*update
EH01	Hospital	Alarm (init.)	Initial situation change	Yes, initialize
EH02	Hospital	Link corridor blockage	Edge weight ∞	Yes, LPA*update
EH03	Hospital	Ward fire	Node and edges ∞	Yes, LPA*update
ES01	School	Alarm (init.)	Initial situation change	Yes, initialize
ES02	School	Wing corridor blockage	Edge weight ∞	Yes, LPA*update
ES03	School	Junction obstruction	Edge weight 3$\times$	Yes, LPA*update
ES04	School	Main hall crowding	Edge weight 3$\times$	Yes, LPA*update

Events fall into two categories:

1. **Impact events**: These directly affect reachability or cost in the graph. Examples: a door is locked (blocking an edge), a staircase becomes congested (increasing edge cost by 3$\times$), a room fills with smoke (rendering it unreachable). Impact events always trigger re-planning.

2. **Non-impact events**: These do not directly alter the graph but may affect the planning context (e.g., human behavior, resource availability). In

the current implementation, non-impact events do not trigger re-planning; the system continues with the current plan. However, the architecture accommodates future extension to handle such events via learned policies or domain-specific heuristics (deferred to Chapter 5).

4.5.2 Incremental Re-planning Procedure

When an impact event occurs during evacuation execution, the system executes the following procedure:

1. **Event Detection**: The environment monitoring system detects a change (e.g., a sensor indicates a door is blocked) and raises an event notification with the event ID, timestamp, and affected location/entity.

2. **Situation Update**: The knowledge base is updated to reflect the new condition. For example, if a corridor is blocked, the corresponding SWRL rule body condition (e.g., `corridor_clear`(c)) is asserted to be false, or edge weights in the ontology are modified.

3. **Graph Invalidation**: The set of affected nodes and edges is determined by querying the ontology. For a blocked corridor c, any situation instance s whose definition depends on the accessibility of c is marked as potentially invalid. Similarly, any edge transition involving c has its weight updated.

4. **LPA* Update**: For each affected edge, the LPA*algorithm executes its update step:

   (a) Retrieve the predecessor node n' and successor node n of the updated edge.

   (b) Recompute $\text{rhs}(n) = \min_{n'' \in \text{pred}(n)} \{g(n'') + w(n'', n)\}$.

   (c) If $\text{rhs}(n) \neq g(n)$, insert n into the priority queue if not already present, or update its priority.

   (d) Continue expansion from the queue until a new locally consistent path is found (i.e., $g(\text{goal}) = \text{rhs}(\text{goal})$).

5. **Path Relay**: Once a new plan is computed (or the existing plan is validated as still optimal), the system extracts the next action and relays it to the execution layer (e.g., person moves from current location to next location along the new path).

6. **Repeat**: Continue monitoring and repeat steps 1–5 until evacuation completion.

The key efficiency gain is that LPA* does not recompute the entire path from scratch; it only re-expands nodes whose rhs values changed, often leaving large portions of the previous search tree intact. In semi-dynamic environments (where events are sparse), this can be orders of magnitude faster than full re-planning.

4.5.3 Backtracking Mechanism

Incremental re-planning assumes that a new path can be found from the current location to the goal. However, in some cases, a situation instance becomes a dead-end: all possible transitions from that situation lead either to invalid successors, to situations with infinite cost, or to cyclic loops. In such cases, the system cannot proceed forward and must *backtrack*.

The reactive backtracking mechanism operates as follows:

1. **Dead-End Detection**: During plan execution, if the planner attempts to expand the current situation and finds that all successors are either invalid, infinitely costly, or already visited, the current situation is declared a dead-end.

2. **Rollback**: The system rewinds the execution path to the last situation instance that had multiple feasible successors (a *branching node*). This is achieved by maintaining a stack of parent pointers; backtracking pops the stack until a branching node is found.

3. **Edge Marking**: The edge that led to the dead-end is marked as locally blocked (cost set to infinity). This prevents the planner from exploring the same dead-end again in subsequent iterations.

4. **Re-planning**: From the branching node, LPA* is invoked to compute an alternative path that avoids the newly-blocked edge. If multiple alternatives exist, the optimal one (according to f-score) is selected.

5. **Execution Resume**: Execution resumes from the branching node along the new path.

Importantly, the backtracking mechanism in this chapter is *reactive*: it responds to dead-ends that have already occurred during execution. In Chapter 5, a *predictive* backtracking mechanism is introduced, where the BiLSTMmodel anticipates potential dead-ends before they occur and proactively avoids them. The reactive mechanism serves as the deterministic baseline.

Example: Suppose in the simple scenario above, after the person reaches $s_2 =$ `PersonAtStaircase`(alice, sc), a fire event blocks the staircase, making rule R3 (PersonAtStaircase $\rightarrow$ PersonOnStaircase) infeasible. The planner detects this dead-end. It backtracks to $s_1 =$ `PersonInCorridor`(alice, c), marks the edge $s_1 \rightarrow s_2$ as blocked, and replans. If another corridor leads to an alternative staircase, the new path uses that route. If no alternative exists, the system backtracks further to s_0 and may re-plan to a different corridor.

4.6 Scenario Taxonomy

To systematically evaluate the planning and re-planning algorithms, we define a taxonomy of evacuation scenarios spanning three complexity tiers. Each tier tests different aspects of the planner: baseline path quality, incremental re-planning efficiency, and backtracking robustness.

Table 4.2 summarizes the taxonomy:

Table 4.2: Scenario Taxonomy: Tier Distribution, Events, and Computational Characteristics

Tier	**Count (per domain)**	**% of Total**	**Key Events**
Simple	15/50	30%	1 alarm (t=0), no invalidation
Semi-Complex	20/50	40%	Alarm + 1–2 obstructions (staggered)
Complex	15/50	30%	Alarm + 2–3 obstructions, $\geq$ 1 dead-end

4.6.1 Simple Planning Scenario

Scenario definition: A single alarm event occurs at $t = 0$, placing the building into evacuation mode. No subsequent path invalidation or congestion events occur. The environment remains static throughout execution.

Characteristics:

- Only the initial situation is updated in response to the alarm.

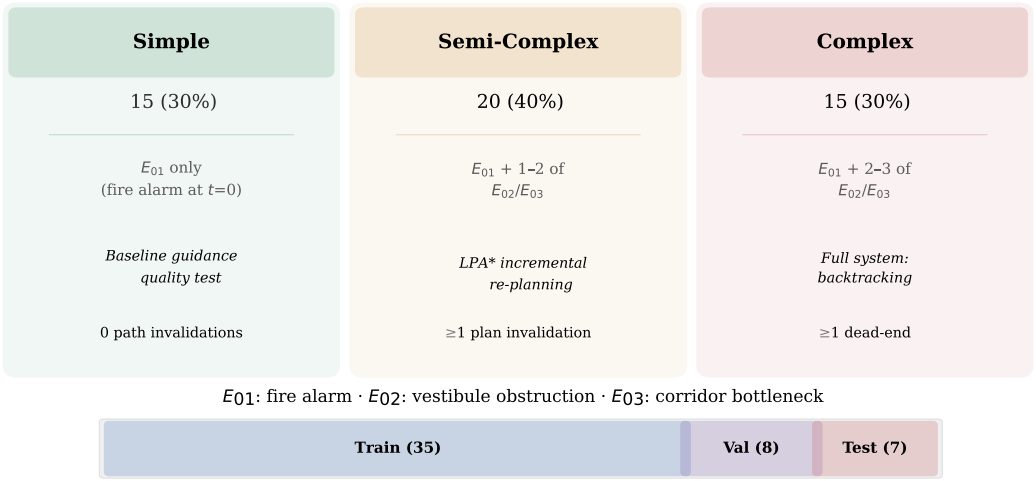

Figure 4.1: Three-tier scenario taxonomy and training–evaluation split for the 50 dynamic evacuation scenarios. Event types: E_{01} (fire alarm), E_{02} (vestibule obstruction), E_{03} (corridor bottleneck). The scenario-level split prevents trajectory leakage between training and evaluation sets.

- Static path planning (Section 4.4) computes the optimal evacuation route.
- No re-planning or backtracking is required.
- Represents 30% (15 of 50) of scenarios in each domain.

Test objective: Validate that the semantic heuristic $h_o(n)$ guides search efficiently to the goal, and that the returned path is optimal in cost. This is the baseline against which dynamic algorithms are measured.

Exemplar: The worked example in Section 4.4.2 (simple two-floor building with one alarm, no obstructions) is a simple scenario. The planner executes a single A* search and returns a path of cost 6.0.

4.6.2 Semi-Complex Planning Scenario

Scenario definition: An alarm event occurs at $t = 0$. Subsequently, one to two path invalidation events are triggered at staggered times (e.g., $t = 5$ and $t = 15$ seconds into execution). These events alter edge weights (congestion) or block edges entirely.

Characteristics:

- The initial plan computed at $t = 0$ may become suboptimal or infeasible at $t = 5$ or $t = 15$.

- The re-planning procedure (Section 4.5.2) is invoked once or twice.
- LPA*'s incremental machinery is tested: only affected nodes are re-expanded, and the previous search tree is partially reused.
- No dead-ends occur; alternative paths exist around any obstruction.
- Represents 40% (20 of 50) of scenarios in each domain.

Test objective: Measure the wall-clock time savings of incremental re-planning versus full re-planning from scratch. Verify that the final path remains optimal and that the system gracefully recovers from local obstructions.

Exemplar: An extension of the simple scenario where, at $t = 5$, a fire event in the primary staircase raises the cost of rule R3 to infinity. The planner (still at s_1, the corridor) detects this via monitoring. LPA*updates the successor costs, finds that the originally-optimal path is now infeasible, and replans. If the scenario allows an alternative staircase reachable from a different corridor, the planner finds that path. The re-plan incurs only the cost of re-expanding affected nodes, not a full search.

4.6.3 Complex Planning Scenario

Scenario definition: An alarm occurs at $t = 0$. Two to three path invalidation events occur at staggered times. At least one of these events creates a dead-end, such that backtracking is required to escape the infeasible branch.

Characteristics:

- Multiple re-planning cycles occur as events are detected.
- Backtracking is triggered when the person reaches a situation from which no feasible forward progress is possible (all successors blocked or infinite-cost).
- The system rolls back to the last branching node, marks the blocking edge, and replans around it.
- May require multiple rounds of re-planning and backtracking before a feasible path is found.
- Represents 30% (15 of 50) of scenarios in each domain.

Test objective: Validate the correctness and efficiency of the backtracking mechanism under adversarial conditions. Measure the overhead of repeated backtracking and re-planning relative to scenarios with simpler event patterns.

Exemplar: In the simple scenario, suppose two events occur: at $t = 5$, the primary staircase becomes congested (cost $\times 3$). At $t = 12$, while the person is navigating the congested staircase, a second event blocks an alternative exit corridor. The person reaches the foot of the stairs (s_2) and attempts to ascend; however, all further progress (via R3) leads to a situation from which exits are blocked. This is detected as a dead-end. The system backtracks to s_1 (the corridor), marks the staircase edge as blocked, and replans. If no other staircase is reachable from s_1, the system backtracks further to s_0 (the room), replans to find a completely different egress route (if one exists), and execution resumes.

4.7 Summary

This chapter has presented a complete framework for semantic heuristic search and event-driven re-planning in ontology-guided emergency evacuation. The key contributions are:

1. **Ontology-Guided Graph Search**: We adapted Lifelong Planning A* to operate over situation instances and SWRL rule instantiations, enabling efficient path planning on dynamically-inferred state spaces that would be prohibitively large to enumerate explicitly.

2. **Semantic Heuristic** $h_o(n)$: We defined a computationally efficient, admissible heuristic that exploits situation type hierarchies and rule chains. The heuristic guides search toward the goal while guaranteeing optimal path discovery. We proved admissibility under both static and dynamic conditions.

3. **Static Planning Baseline**: We established a baseline algorithm for computing evacuation routes under stable conditions, validating the heuristic's effectiveness before dynamic scenarios are introduced.

4. **Incremental Re-planning**: We implemented an event-driven architecture that detects edge cost updates or transition invalidation, updates the graph incrementally, and replans only affected portions of the path. This yields

significant computational savings compared to full re-planning in semi-dynamic scenarios.

5. **Reactive Backtracking**: We introduced a mechanism for recovering from dead-ends by rolling back to the last branching node, marking blocking edges, and re-planning around them. This ensures the system does not give up in the face of local path invalidation.

6. **Scenario Taxonomy**: We defined a three-tier taxonomy (simple, semi-complex, complex) that characterizes the difficulty and computational profile of evacuation scenarios, providing a structured basis for evaluation in Chapter 6.

The algorithms and heuristics presented in this chapter are deterministic and do not incorporate learning. The semantic heuristic $h_o(n)$ is hand-crafted based on domain knowledge and logical inference. In Chapter 5, we extend this framework by introducing a learned heuristic $h_l(n)$ that is trained on evacuation trajectories and combined with $h_o(n)$ via a hybrid function $h'(n) = \min\{h_o(n), (1-\lambda) \cdot h_o(n) + \lambda \cdot h_l(n)\}$, where λ is a blending parameter. The hybrid heuristic leverages learning to further improve search efficiency and robustness.

Throughout this chapter, we have maintained the critical distinction between the *symbolic search layer* (presented here) and the *learning layer* (deferred to Chapter 5). This separation of concerns allows each layer to be understood, validated, and evaluated independently, and facilitates the systematic integration of learning into the symbolic planning framework.

5 BiLSTM-Guided Neuro-Symbolic Planning

5.1 Introduction

The preceding chapter established an ontology-guided incremental search framework in which LPA* operates over a state-transition graph derived from OWL/SWRL knowledge bases, guided by the semantic heuristic $h_o(n)$. That framework achieves sound, admissible planning with event-driven re-planning and reactive backtracking. Two limitations, however, remain. First, the semantic heuristic is computed from type-level ontology structure: it counts the minimum SWRL rule chain length to the goal and weights transitions by average base costs. While admissible, this estimate does not adapt to the dynamic conditions of a particular evacuation scenario,it cannot learn, for instance, that a corridor frequently becomes congested at a specific phase of the evacuation. Second, the reactive backtracking mechanism described in Chapter 4 triggers reversal only after a dead-end is physically encountered, by which point the occupant has already wasted traversal steps along a doomed branch.

This chapter introduces the BiLSTM-guided extension that addresses both limitations. A Bidirectional Long Short-Term Memory network, trained offline on ontology-consistent evacuation trajectories, augments the symbolic planner with two capabilities: (i) remaining cost estimation, which refines the semantic heuristic into a hybrid heuristic $h'(n)$ that can reduce node expansion without sacrificing admissibility; and (ii) backtracking probability prediction, which enables predictive path reversal before a dead-end is reached. Crucially, the learned component does not modify the admissible transition set governed by the ontology and SWRL rules,it affects only the *ordering* of node expansions during search. The symbolic layer (ontology, SWRL rules, semantic heuristic) remains fully independent and can operate without the BiLSTM when $\lambda = 0$.

The chapter is organised as follows. Section 5.2 presents the BiLSTM architecture, beginning with LSTM foundations and progressing to the dual-head design. Section 5.3 formalises the remaining cost estimation head and its loss function.

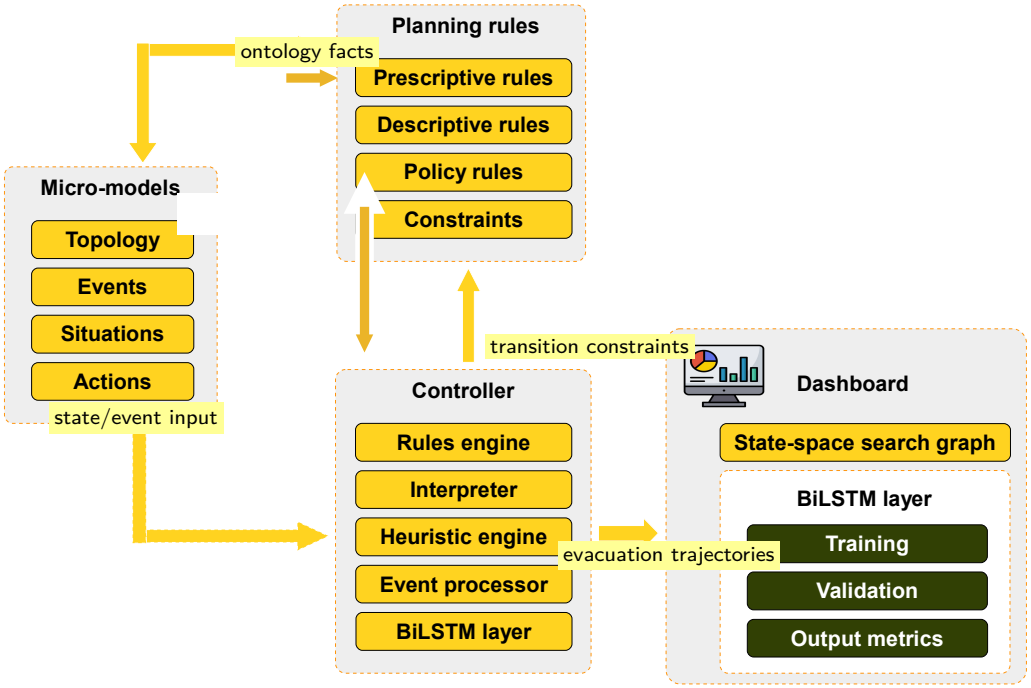

Figure 5.1: Proposed evacuation process of the BiLSTM-guided neuro-symbolic semantic A* planning framework. Yellow labels indicate the type of relationship carried by each connection: ontology facts flow from the micro-models to the planning rules; state and event data feed the controller; transition constraints derived from SWRL rules govern the controller's rules engine; and evacuation trajectories are output to the dashboard and BiLSTM training layer.

Section 5.4 defines the backtracking prediction head, including the threshold parameter τ and its calibration. Section 5.5 derives the hybrid heuristic $h'(n)$ and proves that it preserves admissibility. Section 5.6 describes how the BiLSTM integrates with LPA* and the ontology controller within the neuro-symbolic stack. Section 5.7 details the training data generation pipeline, feature encoding, and hyperparameters. Section 5.8 provides a conceptual comparison of purely semantic and BiLSTM-guided planning. Section 5.9 summarises the chapter.

5.2 BiLSTM Architecture for Evacuation

5.2.1 LSTM Foundations

Evacuation planning in dynamic environments requires a model that can reason over variable-length sequences of past observations,occupant movements, event triggers, cost changes,while selectively retaining information relevant to future decisions. Standard recurrent neural networks suffer from vanishing and exploding gradients when trained on long sequences, limiting their ability to capture dependencies spanning many time steps. The Long Short-Term Memory (LSTM) architecture [40] addresses this limitation through a gated memory cell that regulates information flow across time steps.

At each time step t, an LSTM cell receives an input vector $\mathbf{x}_t$ (representing the current evacuation features: situation type, active events, available actions) and the previous hidden state $\mathbf{h}_{t-1}$. Three gates,forget, input, and output,control how information is stored, updated, and emitted. The cell update equations are:

$$\begin{cases} \mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f), \\ \mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \\ \tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c), \\ \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\ \mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o), \\ \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \end{cases} \tag{5.1}$$

where σ denotes the logistic sigmoid function, $\odot$ denotes element-wise multiplica-

tion, $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_c, \mathbf{W}_o$ are learnable weight matrices, and $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o$ are bias vectors.

The forget gate $\mathbf{f}_t$ determines which elements of the previous cell state $\mathbf{c}_{t-1}$ to retain. In the evacuation context, this allows the model to discard information about spaces that are no longer reachable (e.g. a floor that has been fully evacuated) while preserving information about active hazards. The input gate $\mathbf{i}_t$ controls how much of the candidate update $\tilde{\mathbf{c}}_t$ is written to the cell state, enabling the model to incorporate new observations,such as a freshly triggered event,selectively. The output gate $\mathbf{o}_t$ regulates the hidden state $\mathbf{h}_t$ that is passed to subsequent layers and to the prediction heads. This gated architecture enables the LSTM to maintain a compact, task-relevant summary of the entire movement history $\mathcal{H}$ as the occupant traverses the state-transition graph.

5.2.2 Bidirectional Extension

A unidirectional LSTM processes the movement history $\mathcal{H} = (s_0, a_0, s_1, a_1, \ldots, s_t)$ in chronological order, producing a hidden state $\overrightarrow{\mathbf{h}}_t$ that summarises the trajectory up to time t. This forward pass captures causal dependencies: the model learns which sequences of situations and actions tend to precede high remaining cost or imminent dead-ends.

However, evacuation trajectories also exhibit informative reverse-order structure. The sequence of actions *from the goal backwards* encodes the difficulty of the remaining path: a trajectory whose final segments involve multiple staircase descents and corridor traversals is structurally different from one that terminates with a direct exit. A backward LSTM, processing $\mathcal{H}$ in reverse chronological order, produces a hidden state $\overleftarrow{\mathbf{h}}_t$ that captures this complementary perspective.

A Bidirectional LSTM [30, 78] combines both directions by concatenating the forward and backward hidden states:

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t \parallel \overleftarrow{\mathbf{h}}_t], \tag{5.2}$$

where $\parallel$ denotes vector concatenation. The combined representation $\mathbf{h}_t \in \mathbb{R}^{2d}$ (with d hidden units per direction) provides the prediction heads with a unified encoding that captures both the causal history leading to the current situation and the structural characteristics of the trajectory as a whole.

The bidirectional architecture is particularly well suited to evacuation planning

for two reasons. First, during training, complete trajectories are available (since training data are generated by the symbolic planner running to completion), so the backward pass has access to the full future context. Second, at inference time, the backward pass operates over the movement history accumulated so far: although the true future trajectory is unknown, the backward processing of the *observed* partial trajectory still provides useful structural features (e.g. whether the occupant has recently descended multiple floors, which correlates with proximity to exit).

5.2.3 Dual-Head Architecture

The BiLSTM encoder produces a single hidden representation $\mathbf{h}_t$ from the movement history $\mathcal{H}$. Two task-specific prediction heads branch from this shared representation, each implemented as a two-layer fully connected network:

1. **Cost estimation head.** A regression head that predicts the remaining cost $h_l(n)$ from the current situation n to the nearest goal (any `person_outside_building` situation). The architecture is Dense(64) $\rightarrow$ Dense(1, linear). The output is a non-negative scalar representing the estimated number of state transitions remaining.

2. **Backtracking prediction head.** A binary classification head that predicts the probability $p_{\text{bt}}(n)$ that the occupant will need to backtrack within the next $k = 3$ steps. The architecture is Dense(64) $\rightarrow$ Dense(1, sigmoid). The output is a probability in $[0, 1]$.

The shared BiLSTM encoder enables both heads to benefit from the same temporal representation without requiring separate feature extraction pipelines. The two heads serve complementary purposes within the planner: the cost head refines node expansion ordering (via the hybrid heuristic $h'(n)$), while the backtracking head triggers predictive path reversal (via the threshold τ). These two decisions are architecturally independent,the hybrid heuristic governs *which* node to expand next, while the backtracking threshold governs *whether* to reverse course,but both draw on the same learned temporal representation.

The shared-encoder, dual-head design offers a practical advantage: the BiLSTM is invoked once per decision point, and both outputs are available simultaneously. This avoids the computational overhead of maintaining two separate sequence

models and ensures that the cost and backtracking predictions are conditioned on an identical temporal context.

5.3 Remaining Cost Estimation Head

5.3.1 Problem Formulation

The cost estimation head approximates the optimal remaining cost from a situation n to the nearest goal. Formally, let $h^*(n)$ denote the true minimum cost from n to any goal situation under the current edge weights. The learned heuristic $h_l(n)$ is defined as the output of the cost head given the movement history $\mathcal{H}$ augmented with the candidate node n:

$$h_l(n) = \text{BiLSTM\_cost\_head}(\mathcal{H} \cup \{n\}). \tag{5.3}$$

The movement history $\mathcal{H}$ provides temporal context that is unavailable to the purely structural semantic heuristic $h_o(n)$. Specifically, $\mathcal{H}$ encodes: (i) which spaces the occupant has already traversed, revealing the topology explored so far; (ii) which events have been encountered, providing information about the current hazard state; and (iii) the sequence of actions taken, capturing patterns (such as repeated re-planning or corridor traversals) that correlate with longer remaining paths. The semantic heuristic $h_o(n)$ computes a static, type-level estimate based on the ontology's class hierarchy; the learned heuristic $h_l(n)$ complements this with an instance-level, history-conditioned estimate.

The cost head produces a non-negative output by applying a clipping operation $h_l(n) \leftarrow \max(0, h_l(n))$ at inference time, ensuring that the learned estimate does not produce a negative heuristic value. This is a necessary precondition for the admissibility-preserving blend described in Section 5.5.

5.3.2 Loss Function and Training Objective

The cost head is trained to minimise the mean squared error (MSE) between the predicted remaining cost and the true remaining cost computed from the training

trajectories:

$$\mathcal{L}_{\text{cost}} = \frac{1}{M} \sum_{j=1}^{M} (\hat{c}_j - c_j)^2, \tag{5.4}$$

where M is the number of training decision points, $\hat{c}_j$ is the predicted remaining cost for decision point j, and c_j is the true remaining cost (measured as the actual number of state transitions from the decision point to `person_outside_building` in the training trajectory).

MSE is chosen over alternatives such as mean absolute error (MAE) because it penalises large prediction errors more heavily. In the evacuation context, a large overestimate of remaining cost could cause the planner to deprioritise a node that is actually close to the goal, while a large underestimate could cause premature commitment to a suboptimal branch. The quadratic penalty of MSE ensures that the model is trained to avoid both extremes.

The true remaining cost c_j is derived from the symbolic planner's own execution traces (running at $\lambda = 0$, i.e. pure semantic LPA*), not from an external ground truth. This means the BiLSTM learns to approximate the symbolic planner's cost function in a data-driven manner. The advantage is that the training labels are fully reproducible from the ontology and the planner; the limitation is that the learned cost estimate inherits any systematic biases of the $\lambda = 0$ planner (e.g. if the semantic heuristic consistently overestimates cost on certain graph topologies, the training labels will reflect this). This closed-loop characteristic is explicitly acknowledged: the BiLSTM refines the symbolic planner's estimates; it does not provide an independent ground truth.

5.4 Backtracking Prediction Head

5.4.1 Binary Classification Formulation

The backtracking head predicts the probability that the occupant will need to reverse direction within a short lookahead horizon. This is formulated as a binary classification problem. For each decision point n in a training trajectory, the binary label $y_{\text{bt}} \in \{0, 1\}$ is defined as:

$$y_{\text{bt}}(n) = \begin{cases} 1, & \text{if the occupant backtracks within the next } k \text{ steps,} \\ 0, & \text{otherwise,} \end{cases} \tag{5.5}$$

where $k = 3$ is the backtracking lookahead horizon. A backtracking event is identified in the training data when the occupant's trajectory includes a reversal: a transition to a previously visited situation within the k-step window.

The backtracking head outputs a probability $p_{\text{bt}}(n) \in [0, 1]$ via a sigmoid activation:

$$p_{\text{bt}}(n) = \sigma(\mathbf{w}_{\text{bt}}^{\top}\mathbf{z}_{\text{bt}} + b_{\text{bt}}), \tag{5.6}$$

where $\mathbf{z}_{\text{bt}}$ is the output of the penultimate dense layer in the backtracking head, and $\mathbf{w}_{\text{bt}}, b_{\text{bt}}$ are the final layer's weight vector and bias.

The backtracking head is trained using binary cross-entropy with class weighting to address the imbalance between backtracking and non-backtracking decision points:

$$\mathcal{L}_{\text{bt}} = -\frac{1}{M}\sum_{j=1}^{M}\Big[w_1\, y_j \log p_j + w_0\,(1 - y_j)\log(1 - p_j)\Big], \tag{5.7}$$

where w_1 and w_0 are class weights inversely proportional to class frequency. The backtrack-positive rate in the Tower Building training data is approximately 30%, making the non-backtracking class roughly twice as prevalent. Class weighting prevents the model from trivially predicting the majority class and ensures adequate sensitivity to the minority (backtracking) class.

5.4.2 Prediction Threshold and Decision Logic

The continuous probability $p_{\text{bt}}(n)$ is converted to a binary decision by the threshold parameter $\tau \in [0, 1]$:

$$\text{backtrack\_decision}(n) = \begin{cases} 1, & \text{if } p_{\text{bt}}(n) \geq \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

When backtrack_decision$(n) = 1$, the planner initiates predictive backtracking: it reverses along the already-traversed path to the most recent branching node, marks the forward edge as locally blocked ($w = \infty$), and re-invokes LPA* from the branching node. This mechanism enables the planner to abandon a doomed branch *before* reaching a dead-end, saving the traversal steps that would otherwise be wasted.

The threshold τ governs the sensitivity–specificity trade-off of the backtracking mechanism:

- A low τ (e.g. 0.3) produces aggressive backtracking: more frequent early reversals that avoid dead-ends but risk unnecessary detours when the prediction is a false positive.
- A high τ (e.g. 0.8) produces conservative backtracking: fewer reversals with fewer false positives, but the planner may reach dead-ends that could have been anticipated.
- The standard decision boundary $\tau = 0.5$ treats the two error types symmetrically.

The optimal threshold τ^* is calibrated per building on the validation set (15% of scenarios, held out from training) using Receiver Operating Characteristic (ROC) analysis. The calibration procedure sweeps τ from 0.0 to 1.0 in increments of 0.01 and selects τ^* using Youden's J statistic:

$$\tau^* = \arg\max_{\tau} \Big[\mathrm{TPR}(\tau) - \mathrm{FPR}(\tau) \Big], \tag{5.9}$$

where $\mathrm{TPR}(\tau)$ is the true positive rate (proportion of actual backtracking situations correctly predicted) and $\mathrm{FPR}(\tau)$ is the false positive rate (proportion of non-backtracking situations incorrectly flagged). Youden's J maximises the vertical distance from the diagonal (random classifier) on the ROC curve, providing the optimal balance between sensitivity and specificity for the binary backtracking decision.

The threshold is calibrated independently for each building because the backtracking class distribution and graph topology differ across buildings. The Tower Building, with its single exit and predominantly vertical evacuation, exhibits a moderate backtrack-positive rate; the Hospital, with its dual-block layout and multiple exits, presents different backtracking patterns; and the School, with four exits and a hub-and-wing topology, may require a different sensitivity level. Per-building calibration ensures that the threshold is tuned to the specific characteristics of each environment.

A sensitivity analysis is conducted alongside calibration, sweeping τ over $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ and reporting T_{avg} as a function of τ. This analysis demonstrates the robustness of the planner to the exact threshold value and verifies that the Youden-optimal τ^* falls within a stable performance region. The interaction between τ and λ is addressed empirically in Chapter 6: while the two parameters are architecturally independent (they govern different decisions,node ordering and path reversal,

respectively), there may be an empirical interaction through the quality of p_{bt} predictions at different λ values.

5.5 Hybrid Heuristic Function

5.5.1 Design: Bounded Convex Blend

The semantic heuristic $h_o(n)$ and the learned heuristic $h_l(n)$ are combined into a hybrid heuristic $h'(n)$ using a bounded convex blend controlled by the parameter $\lambda \in [0, 1]$:

$$h'(n) = \min\Big\{h_o(n),\ (1-\lambda)\, h_o(n) + \lambda\, h_l(n)\Big\}, \quad 0 \leq \lambda \leq 1. \tag{5.10}$$

This formulation has three important properties:

Admissibility preservation. The outer min operator ensures that $h'(n) \leq h_o(n)$ for all nodes n and all values of λ. Since $h_o(n)$ is admissible (Proposition 4.1 in Chapter 4), $h'(n)$ is also admissible. The formal proof is given in Section 5.5.2.

Graceful degradation. When $\lambda = 0$, the blend reduces to $h'(n) = \min\{h_o(n), h_o(n)\} = h_o(n)$, recovering pure semantic LPA*. This means the framework can be deployed on a new building immediately after ontology instantiation, without requiring any BiLSTM training data. Learned guidance is introduced incrementally as training trajectories become available.

Bounded neural influence. When $h_l(n) < h_o(n)$,i.e. the learned model predicts that the remaining cost is lower than the semantic estimate,the convex blend produces a value below $h_o(n)$, and the min selects this tighter estimate. This tightens the heuristic, improving search efficiency by directing LPA* towards more promising nodes. When $h_l(n) \geq h_o(n)$, the convex blend exceeds $h_o(n)$, and the min falls back to $h_o(n)$. The learned component can therefore only *reduce* the heuristic estimate relative to the semantic baseline; it can never increase it.

The parameter λ controls the strength of the learned guidance. At $\lambda = 0.3$ (low blend), the convex combination is $0.7 \cdot h_o(n) + 0.3 \cdot h_l(n)$, providing moderate neural influence. At $\lambda = 0.7$ (high blend), the combination is $0.3 \cdot h_o(n) + 0.7 \cdot h_l(n)$, giving the learned estimate dominant weight. The evaluation in Chapter 6 tests both settings alongside $\lambda = 0.0$ (ablation baseline) to characterise the effect of

increasing neural influence on evacuation efficiency, search effort, and backtracking frequency.

5.5.2 Admissibility Preservation Proof

The following proposition establishes that the hybrid heuristic preserves the admissibility of the underlying semantic heuristic and, consequently, the optimality guarantees of LPA*.

Proposition 5.1 (Admissibility of $h'(n)$)**.** *If $h_o(n)$ is an admissible heuristic,i.e. $h_o(n) \leq h^*(n)$ for all nodes n, where $h^*(n)$ is the true optimal cost to the nearest goal,then $h'(n) \leq h^*(n)$ for all n and all $\lambda \in [0, 1]$.*

Proof. By Eq. (5.10), $h'(n) = \min\{h_o(n),\ (1-\lambda)\, h_o(n) + \lambda\, h_l(n)\}$.

Since $h'(n)$ is defined as the minimum of two quantities, it follows that:

$$h'(n) \leq h_o(n).$$

By hypothesis, $h_o(n) \leq h^*(n)$. Therefore:

$$h'(n) \leq h_o(n) \leq h^*(n),$$

which establishes that $h'(n)$ is admissible. □ □

The proof is independent of the accuracy or behaviour of $h_l(n)$: regardless of whether the learned heuristic overestimates, underestimates, or is arbitrarily inaccurate, the outer min ensures that $h'(n)$ never exceeds the admissible semantic heuristic $h_o(n)$. This architectural guarantee is critical because it means that the BiLSTM cannot compromise the optimality properties of the planner. Even if the learned model produces poor predictions (e.g. on an out-of-distribution building topology), the worst case is that $h'(n) = h_o(n)$ for all nodes, and the planner degrades gracefully to purely semantic search.

The admissibility of $h'(n)$ ensures that LPA* using $f(n) = g(n)+h'(n)$ finds optimal paths on the state-transition graph under the current edge weights. Combined with the incremental update property of LPA* (which preserves optimality across edge cost changes), this guarantees that the neuro-symbolic planner maintains the same correctness properties as the purely symbolic variant while potentially expanding fewer nodes.

5.6 Integration with LPA* and Ontology Controller

5.6.1 Neuro-Symbolic Stack Architecture

The complete neuro-symbolic evacuation planning framework comprises seven interacting components, organised into a layered architecture. Table 5.1 summarises the role, inputs, and outputs of each component.

Table 5.1: Roles and interactions of the main components in the neuro-symbolic evacuation stack.

Component	**Role and responsibility**
OWL/RDF ontology	Consumes building entities, occupants, and hazards to produce semantic situations and facts. Defines the semantic state space and typed entities on which all planning operates.
SWRL rules + constraints	Consumes ontology facts and event assertions to produce admissible transitions and pruned neighbours. Acts as the admissibility gate: transitions violating safety or access constraints are rejected at expansion time.
Semantic heuristic $h_o(n)$	Consumes current node, goal, and ontology inferences to produce an admissible remaining-cost estimate. Guides search using ontology-derived structure; must remain admissible.
Planner (Semantic LPA*)	Consumes admissible neighbours, costs, and heuristic to produce a candidate evacuation path. Performs incremental heuristic search using $f(n) = g(n) + h(n)$.
Replanner	Consumes new event assertions and updated ontology to produce a repaired plan. Triggered by events that change costs or invalidate transitions; updates only affected portions.
Backtracker	Consumes execution failure or invalidated transition to produce rollback to last feasible situation plus a re-planning trigger.
BiLSTM guidance module	Consumes history of semantic state–action sequence to produce predicted remaining cost and backtracking likelihood. Does not create actions or bypass SWRL admissibility; influences only node expansion ordering.
Controller	Consumes events, reasoning outputs, and planner outputs to coordinate action execution, re-planning triggers, and backtracking. Acts as the runtime glue between symbolic and learned modules.

A critical architectural property is the **independence of the symbolic and learned layers**. The ontology, SWRL rules, and semantic heuristic $h_o(n)$ operate without any dependency on the BiLSTM. Conversely, the BiLSTM consumes

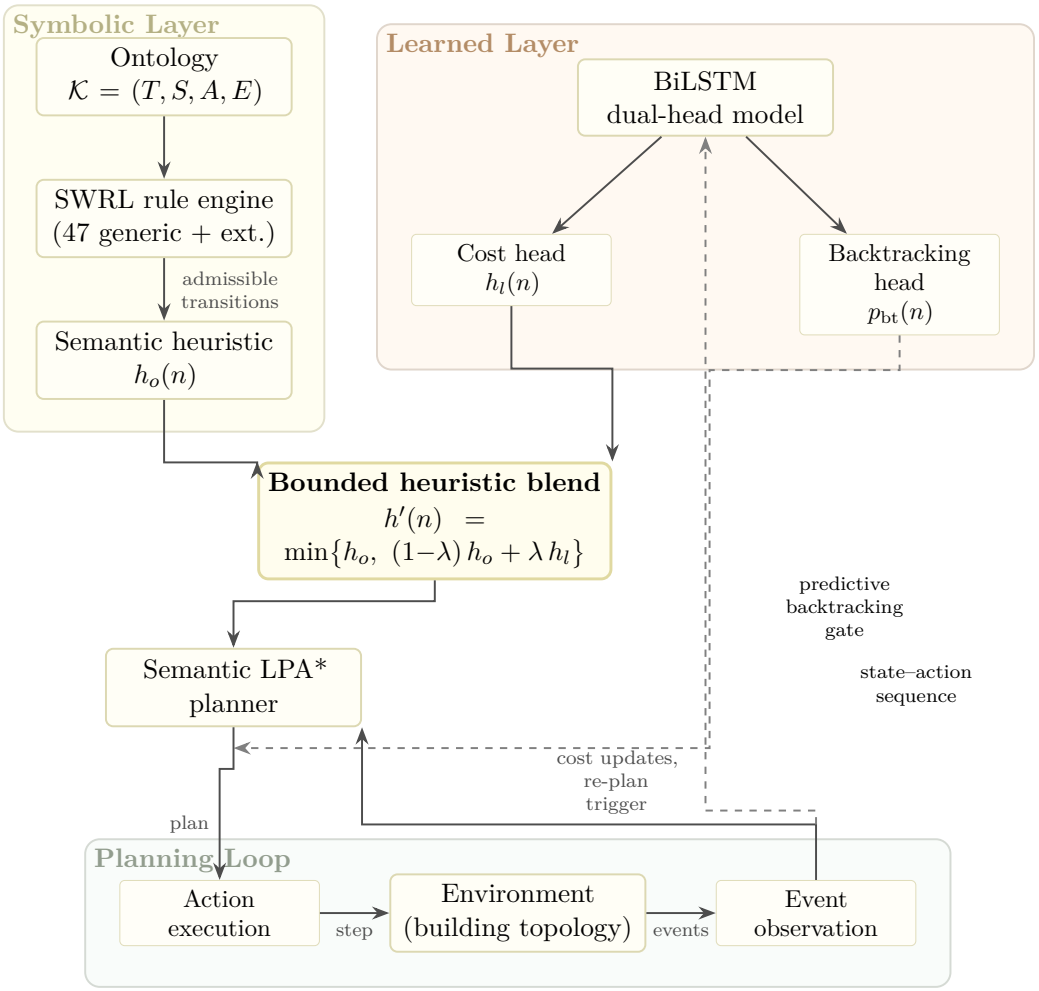

Figure 5.2: System architecture of the neuro-symbolic evacuation planning framework. The symbolic layer (ontology, SWRL rules, semantic heuristic) operates independently of the learned layer (BiLSTM dual-head model). The two layers interact only through the bounded heuristic blend $h'(n)$; the backtracking head provides a predictive gate that does not modify admissible transitions. The planning loop exchanges actions and event observations with the building environment.

symbolic state–action sequences but does not modify the ontology or the admissible transition set. The two layers interact only through the bounded blend in Eq. (5.10). This independence enables incremental deployment: setting $\lambda = 0$ recovers pure semantic LPA*, which can be deployed immediately on a new building once the ontology is instantiated, before any BiLSTM training data have been generated. Learned guidance is introduced incrementally as training trajectories become available, with larger λ values increasing the influence of the learned component without violating optimality or policy constraints.

5.6.2 Heuristic Estimation Using Inferred Paths

The integration of the BiLSTM into the LPA* search loop proceeds in four steps at each node expansion. Given the current node being expanded and a candidate neighbour, the planner:

1. **Computes $h_o(n)$.** The semantic heuristic is evaluated for the candidate neighbour by querying the ontology for inferred paths, vertical navigation requirements, and policy-compliant transitions. This computation uses the procedure described in Chapter 4: BFS over situation type hierarchies, weighted by average base costs per transition type, with a quality factor discount for uncertain inferred routes.

2. **Evaluates $h_l(n)$.** The movement history $\mathcal{H}$, augmented with the candidate neighbour, is passed to the BiLSTM. The model processes the sequence through the bidirectional encoder and returns the cost head output $h_l(n)$. The output is clipped to ensure non-negativity: $h_l(n) \leftarrow \max(0, h_l(n))$.

3. **Computes $h'(n)$.** The bounded convex blend (Eq. (5.10)) combines $h_o(n)$ and $h_l(n)$ using the configured λ value.

4. **Updates $f(n)$.** The evaluation function $f(n) = g(n) + h'(n)$ is computed and used to order the candidate in the LPA* priority queue.

Simultaneously, the backtracking head output $p_{\text{bt}}(n)$ is checked against τ. If $p_{\text{bt}}(n) \geq \tau$, the planner initiates predictive backtracking as described in Section 5.4.2, overriding the normal advance-to-next-node decision.

The evaluation function for the neuro-symbolic planner is therefore:

$$f(n) = g(n) + h'(n), \tag{5.11}$$

where $g(n)$ is the standard shortest-path cost from the start node, maintained incrementally by LPA*. Because $h'(n) \leq h_o(n) \leq h^*(n)$ (Proposition 5.1), admissibility and optimal-path completeness of LPA* are preserved. The BiLSTM influences only the ordering of node expansions: nodes for which the learned model predicts lower remaining cost are expanded earlier, potentially reducing the total number of expansions required to find the optimal path.

5.7 Training Data Generation and Methodology

5.7.1 SWRL Trajectory Generation

The BiLSTM requires supervised training data in the form of complete evacuation trajectories. These trajectories are generated by the ontology-guided planner operating at $\lambda = 0.0$ (pure semantic LPA*) as the data-generating process. The use of the symbolic planner as the trajectory source has two consequences: (i) all training trajectories are ontology-consistent,every transition is a policy-compliant (situation, action, next-situation) triple instantiated from the SWRL rules; and (ii) the BiLSTM learns to approximate the symbolic planner's cost function, not an external or physical ground truth.

The data generation pipeline operates in six stages:

Stage 1: Scenario instantiation. Each scenario is defined by a building ABox, an event sequence drawn from the building's event catalogue, and a set of occupant start positions sampled uniformly over occupied spaces.

Stage 2: Trajectory generation. The pure semantic LPA* planner ($\lambda = 0.0$) is executed on each scenario. For each occupant, the planner records the complete state–action–state sequence from the initial situation to `person_outside_building` (or timeout). Each transition in the sequence corresponds to a SWRL rule instantiation, ensuring ontology consistency.

Stage 3: Feature extraction. Raw trajectories are converted to the BiLSTM input format. Each decision point (a non-terminal state in the trajectory) produces one training example consisting of the movement history up to that point and the associated feature vector.

Stage 4: Label generation. Two labels are computed for each decision point: (i) the remaining cost label, defined as the actual number of state transitions from

the current node to `person_outside_building` in the recorded trajectory; and (ii) the backtracking label, a binary indicator of whether the occupant backtracks within the next $k = 3$ steps.

Stage 5: Train/validation/test split. The 50 scenarios per building are split at the scenario level (not the trajectory level) to prevent data leakage: 35 scenarios (70%) for training, 8 scenarios (16%) for validation, and 7 scenarios (14%) for testing. No trajectory from a training scenario appears in the test evaluation.

Stage 6: Training. The dual-head BiLSTM is trained on the combined loss function (described in Section 5.7.3) using the training set, with early stopping monitored on the validation set.

Table 5.2 summarises the training data configuration and estimated volumes for the Tower Building (the primary evaluation environment from the published work) alongside the Hospital and School buildings for which the same pipeline architecture is applied.

Table 5.2: Training data configuration and estimated volumes per building.

Property	**Tower**	**Hospital**	**School**
Building topology			
State-transition graph nodes	26	∼24	∼32
SWRL rules (generic + extension)	47 + 56	47 + 38	47 + 39
Graph diameter ($d_{\max}$ steps)	∼18	∼14	∼12
Scenario generation			
Total scenarios	50	50	50
Simple / Semi-Complex / Complex	15 / 20 / 15	15 / 20 / 15	15 / 20 / 15
Hazard event types	3	3	4
Modelled occupants	50	40	50
Trajectory statistics (estimated)			
Total raw trajectories	2,500	2,000	2,500
Augmented training samples	∼75,000	∼55,000	∼62,000
Split			
Train / Val / Test scenarios	35 / 8 / 7	35 / 8 / 7	35 / 8 / 7

The trajectory volumes in Table 5.2 are order-of-magnitude estimates derived from the graph structure (graph diameter, number of occupants, augmentation ratio). Exact values depend on the dynamic trajectory lengths produced by the planner under each scenario's event sequence and will be confirmed upon experimental execution.

5.7.2 Feature Encoding and Data Pipeline

Each time step in the movement history is encoded as a fixed-length feature vector comprising seven components:

1. **Current situation type** (one-hot encoded). The ontology situation class of the occupant at this time step (e.g. `person_in_corridor`, `person_on_staircase`). This categorical encoding captures the semantic type of the current space.

2. **Current space identifier** (one-hot encoded). The specific ABox individual representing the physical space (e.g. `Corridor_B_GF`). This captures instance-level identity beyond the type-level situation class.

3. **Floor level** (integer). The current floor, encoded as 0 (ground), 1 (first), and so forth. This feature provides vertical position information that the BiLSTM can correlate with remaining cost (higher floors generally require more transitions to reach ground-level exits).

4. **Steps taken** (integer). The number of state transitions elapsed since evacuation start. This temporal feature helps the model distinguish early-stage trajectories (where many options remain) from late-stage trajectories (where the occupant is close to exit or trapped).

5. **Active events** (binary vector). A binary indicator for each event type in the building's event catalogue (e.g. $[E_{01}, E_{02}, E_{03}]$ for the Tower Building). Active events signal which hazards are currently affecting the environment, providing the model with information about the current state of the edge weight modifications.

6. **Available actions** (binary vector). A binary indicator for each action type that is currently admissible from the occupant's situation, as determined by SWRL rule evaluation. This encodes the local branching structure of the state-transition graph at the current node.

7. **Semantic heuristic value** (float). The value of $h_o(n)$ at the current node. Providing the semantic heuristic as an input feature enables the BiLSTM to learn corrections relative to the ontology-derived estimate rather than learning the cost function from scratch.

Variable-length movement histories are padded to the maximum trajectory length observed in the training set for the relevant building. Padding uses zero vectors, and

the BiLSTM processes the padded sequences using standard masking techniques to ignore padded time steps during the forward and backward passes.

The augmentation strategy uses sliding-window sub-sampling over complete trajectories. For a trajectory of length L, the window generates $L-1$ partial trajectories, each ending one step further along the full trajectory. This increases the effective training set size by a factor proportional to the mean trajectory length, yielding approximately 192,000 augmented samples across all three buildings (Table 5.2).

5.7.3 Hyperparameters and Training Protocol

Table 5.3 summarises the BiLSTM model configuration and training hyperparameters.

Table 5.3: BiLSTM model configuration and training hyperparameters.

Parameter	**Value**
Architecture	
BiLSTM layers	2 (bidirectional)
Hidden units per direction	128
Cost head	Dense(64) $\rightarrow$ Dense(1, linear)
Backtracking head	Dense(64) $\rightarrow$ Dense(1, sigmoid)
Input sequence length	Variable (padded to max trajectory length)
Training	
Optimiser	Adam [50] (lr = 0.001)
Batch size	64
Epochs	100 (early stopping, patience = 10)
Cost loss	Mean squared error (MSE)
Backtracking loss	Binary cross-entropy (class-weighted)
Combined loss	$\mathcal{L} = 0.5 \cdot \mathcal{L}_{\text{cost}} + 0.5 \cdot \mathcal{L}_{\text{bt}}$
Backtracking lookahead k	3 steps

The combined loss function weights the two heads equally ($\alpha_c = \alpha_b = 0.5$):

$$\mathcal{L} = \alpha_c \cdot \mathcal{L}_{\text{cost}} + \alpha_b \cdot \mathcal{L}_{\text{bt}}. \tag{5.12}$$

Equal weighting reflects the comparable importance of the two heads in the planning framework: the cost head influences node expansion ordering (affecting search efficiency), while the backtracking head influences path reversal decisions (affecting dead-end avoidance). Alternative weightings were not explored in the current work but represent a potential avenue for tuning.

The same architectural configuration is used across all three buildings. Only the input and output dimensions change based on the building's state space (number of situation types, number of space individuals, number of event types, number of action types). This design choice tests whether a fixed architecture generalises across different building topologies, with the hypothesis that building-specific adaptation is achieved through building-specific training data rather than architectural modifications.

Early stopping monitors the combined loss on the validation set with a patience of 10 epochs: training halts if the validation loss does not improve for 10 consecutive epochs, and the model weights from the best validation epoch are retained. This prevents overfitting to the training trajectories and ensures that the model generalises to unseen scenarios within the same building.

Training uses fixed random seeds for scenario generation, occupant placement, and BiLSTM weight initialisation to ensure full reproducibility. All hyperparameters are reported in this thesis and in the published manuscript [23].

5.8 Conceptual Comparison: Semantic vs. BiLSTM-Guided Planning

The two planning modes,purely semantic ($\lambda = 0$) and BiLSTM-guided ($\lambda > 0$),differ in their information sources, decision mechanisms, and failure modes. Understanding these differences clarifies both the contribution of the neural component and the conditions under which it provides the greatest benefit.

Information source. The semantic heuristic $h_o(n)$ draws exclusively on the ontology's type-level class hierarchy: it computes the minimum SWRL rule chain from the current situation type to the goal type, weighted by average base costs. This estimate is context-independent,it returns the same value for a given situation type regardless of the scenario's event history or the occupant's trajectory. The learned heuristic $h_l(n)$ draws on the occupant's movement history $\mathcal{H}$, which encodes the specific sequence of situations traversed, events encountered, and actions taken. This provides an instance-level, context-dependent estimate that can reflect dynamic conditions (e.g. that a frequently congested corridor has been avoided).

Adaptivity. Under purely semantic planning, the heuristic does not adapt across planning episodes within a scenario. Each re-planning invocation recomputes

$h_o(n)$ from the (updated) ontology, but the heuristic function itself is stateless. Under BiLSTM-guided planning, the learned heuristic implicitly adapts as the movement history grows: with each additional step, the BiLSTM receives more context and can refine its remaining-cost estimate. This makes the guidance progressively more informed as the evacuation unfolds.

Backtracking behaviour. Purely semantic planning relies on reactive backtracking: the planner discovers a dead-end only when all forward edges have infinite cost, then reverses to the last branching node. BiLSTM-guided planning adds predictive backtracking: the backtracking head flags high-risk branches before the dead-end is reached, enabling earlier reversal and reduced wasted traversal.

Failure modes. The semantic heuristic cannot fail catastrophically: it is admissible by construction and provides a sound lower bound. Its weakness is that it may be a loose bound (far below $h^*(n)$), leading to excessive node expansions. The learned heuristic can produce inaccurate predictions on out-of-distribution inputs (e.g. novel event combinations not seen during training), but the bounded blend (Eq. (5.10)) ensures that such inaccuracies cannot compromise admissibility. The worst case under poor BiLSTM predictions is that $h'(n) = h_o(n)$ for all nodes, recovering purely semantic behaviour.

Deployment requirement. Semantic planning requires only the instantiated ontology and SWRL rules,no training data, no neural model. BiLSTM-guided planning additionally requires a trained model, which in turn requires a sufficient number of completed evacuation trajectories generated by the symbolic planner. The framework's incremental deployment property (set $\lambda = 0$ until training data are available) bridges this requirement.

5.9 Summary

This chapter has presented the BiLSTM-guided extension to the ontology-guided LPA* planning framework introduced in Chapter 4. The key technical contributions are:

1. A **dual-head BiLSTM architecture** that provides both remaining cost estimation and backtracking probability prediction from a shared bidirectional encoder, trained on ontology-consistent evacuation trajectories.

2. A **hybrid heuristic function** $h'(n) = \min\{h_o(n),\ (1-\lambda)\, h_o(n) + \lambda\, h_l(n)\}$

that blends the admissible semantic heuristic with the learned cost estimate. Proposition 5.1 proves that $h'(n)$ preserves admissibility for all $\lambda \in [0, 1]$, ensuring that the optimality guarantees of LPA* are maintained regardless of the accuracy of the learned component.

3. A **predictive backtracking mechanism** governed by the threshold τ, calibrated per building via ROC analysis and Youden's J statistic, that enables the planner to abandon high-risk branches before dead-ends are reached.

4. A **training data pipeline** that generates supervised training examples entirely from the symbolic planner ($\lambda = 0$), ensuring ontology consistency of all training trajectories and enabling building-specific model training.

5. A **neuro-symbolic integration architecture** in which the symbolic layer (ontology, SWRL rules, semantic heuristic) and the learned layer (BiLSTM) are architecturally independent. The BiLSTM influences only node expansion ordering through the bounded heuristic blend; it does not modify admissible transitions, and safety and policy compliance remain fully governed by the symbolic layer.

The framework supports three evaluation configurations: $\lambda = 0.0$ (pure semantic, ablation baseline), $\lambda = 0.3$ (moderate neural influence), and $\lambda = 0.7$ (strong neural influence). Each configuration can operate with the backtracking head enabled or disabled, yielding additional ablation variants (cost-only, backtrack-only, full). Chapter 6 presents the experimental evaluation of these configurations across all three buildings, measuring evacuation efficiency (T_{avg}), search effort (N_{nodes}), success rate (S_r), and BiLSTM prediction quality (RMSE, AUC, accuracy).

6 Experimental Evaluation and Discussion

6.1 Introduction

The preceding chapters established the three pillars of the proposed neuro-symbolic evacuation planning framework: the ontological domain models that encode building structure and evacuation semantics (Chapter 3), the semantic heuristic search and event-driven re-planning architecture that operates over the ontology-derived state-transition graph (Chapter 4), and the BiLSTM dual-head neural network that augments the symbolic planner with learned cost estimation and predictive backtracking (Chapter 5). This chapter presents the experimental evaluation of the complete framework, addressing two overarching questions. First, does the integration of a learned component into the symbolic planning loop produce measurable improvements in evacuation efficiency? Second, how does the proposed system compare against established search algorithms that operate on the same ontology-derived graph?

The evaluation is structured around a comparative experimental design involving five algorithm families yielding seven evaluation configurations (Section 6.2.4). All algorithms operate on the same weighted state-transition graph derived from the building ontology, ensuring that observed performance differences are attributable to algorithmic properties rather than representational variation. Six research hypotheses (Section 6.2.5) guide the analysis, ranging from the contribution of neural heuristic guidance (H1) through the value of incremental re-planning (H2) to the generalisation of framework performance across structurally distinct building topologies (H5).

The chapter is organised as follows. Section 6.2 describes the experimental setup, including the simulation environment, building topology configurations, and algorithm configurations. Section 6.3 defines the evaluation metrics spanning both evacuation efficiency and prediction performance. Section 6.4 presents the scenario design and generation methodology, including the three-tier scenario taxonomy and the training–evaluation separation protocol. Section 6.5 reports the

BiLSTM prediction performance on both the cost estimation and backtracking classification tasks. Section 6.6 presents the internal planner variant comparison across five λ-configurations. Section 6.7 extends the comparison to external baseline algorithms. Section 6.8 provides a scenario-level breakdown of results across the three complexity tiers. Finally, Section 6.9 discusses the findings in the context of related approaches and identifies limitations and threats to validity.

6.2 Experimental Setup

6.2.1 Simulation Environment

All experiments are conducted within a custom ontology-guided simulation environment implemented in Python. The simulator instantiates the building's OWL ontology (TBox and ABox), executes SWRL rules to derive the state-transition graph $G = (V, E, w)$, and manages the closed-loop interaction between the planner and the dynamic environment. At each simulation step, the controller queries the current occupant situation, invokes the planner to determine the next action, applies the action to transition the occupant to a new state, and checks for incoming hazard events that may alter edge costs in G.

The simulation loop implements the event-driven re-planning architecture described in Chapter 4. When a hazard event occurs at time t, the affected edges in the changed-edge set $C(t)$ have their weights updated according to the edge cost model (see Section 6.2.3). Incremental algorithms (Semantic LPA*, Conventional LPA*, D* Lite) propagate these cost changes through their priority queues without re-initialising the search. Non-incremental algorithms (A*) re-compute the entire path from the current position. Dynamic algorithms (D*) propagate cost changes from the goal. This design ensures that re-planning overhead differences between algorithms are faithfully captured in the evacuation time metric.

Each simulation run models a single occupant navigating from a randomly assigned starting situation to any situation of type `person_outside_building`. The starting position is drawn uniformly from the set of interior situation nodes, excluding exit-adjacent nodes. Each scenario specifies a deterministic sequence of hazard events with pre-defined activation times, ensuring reproducibility across algorithm configurations. For each building, 50 scenarios are executed, yielding 50 matched observations per algorithm configuration.

Random seeds are fixed for scenario generation (occupant placement and event sequencing) and for BiLSTM weight initialisation, enabling exact reproduction of results. The simulation environment, scenario definitions, trained models, and evaluation scripts will be made available in a supplementary code repository.

6.2.2 Building Topology Configurations

The evaluation spans three structurally distinct building topologies, each instantiated as a separate OWL ontology sharing the same 47-rule generic SWRL core described in Chapter 3. The three buildings differ in vertical extent, horizontal layout, number of exits, and branching factor of the state-transition graph, providing a diverse testbed for assessing framework generalisability.

Tower Building. A 10-floor academic building with a single external exit at ground level. The state-transition graph contains 26 situation nodes and approximately 100 edges. The topology is predominantly vertical: occupants on upper floors must descend through a shared staircase system, creating a long graph diameter ($d_{\max} \approx 18$ transitions) and limited alternative routing. The Tower Building is the primary evaluation environment, matching the building used in the published manuscript [23], and provides the canonical empirical baseline against which the extended evaluation is calibrated. Three hazard event types are defined: E01 (fire alarm triggering evacuation), E02 (vestibule obstruction blocking a floor-level exit), and E03 (corridor bottleneck increasing traversal cost). The building-specific SWRL extension layer comprises 56 rules.

Hospital Building. A 3-storey NHS-style hospital with a dual-block layout connected by link corridors. The state-transition graph contains approximately 24 situation nodes. The topology features restricted zones (e.g., operating theatres, intensive care), staircase-only vertical evacuation, and multiple exit points ($n_{\text{exit}} = 3$). The shorter graph diameter ($d_{\max} \approx 14$ transitions) and higher branching factor provide a contrasting evaluation context to the Tower. Three hospital-specific hazard event types are defined: EH01, EH02, and EH03, targeting corridor blockages, link-corridor obstructions, and ward-level containment events respectively. The building-specific SWRL extension layer comprises 38 rules. Forty occupants are modelled per scenario.

School Building. A 2-storey DfE-style finger-block secondary school with three classroom wings connected to a central hub corridor. The state-transition graph contains approximately 32 situation nodes, the largest of the three buildings, with four exit points ($n_{\text{exit}} = 4$) providing the highest exit redundancy. The graph diameter is the shortest ($d_{\max} \approx 12$ transitions) due to the shallow vertical extent and direct wing-to-hub connectivity. Four school-specific hazard event types are defined: ES01 through ES04, covering assembly hall congestion, wing corridor blockage, hub obstruction, and staircase bottleneck. The building-specific SWRL extension layer comprises 39 rules, including `go_up` actions for backtracking support. Fifty occupants are modelled per scenario.

Table 6.1 summarises the key topological properties of the three buildings.

Table 6.1: Comparison of building topology, graph structure, and ontology rule counts across the three evaluation buildings (Tower, Hospital, School).

Property	**Tower**	**Hospital**	**School**
Floors	10	3	2
Graph nodes ($\|V\|$)	26	∼24	∼32
Graph diameter ($d_{\max}$)	∼18	∼14	∼12
Exit points (n_{exit})	1	3	4
Hazard event types	3	3	4
Generic SWRL rules	47	47	47
Extension SWRL rules	56	38	39
Modelled occupants	50	40	50

The building models represent deliberate topological abstractions rather than compliance-level architectural surveys. The Tower Building is modelled with a single staircase and a single exit to produce a maximally constrained vertical evacuation topology in which hazard events can create true dead-end conditions requiring backtracking, the primary algorithmic capability under evaluation. Real buildings of comparable scale would typically provide multiple protected escape routes and final exits in accordance with building regulations. The reduced exit counts in the evaluation models intensify the planning challenge and isolate the contribution of the re-planning and backtracking mechanisms; they do not imply that the framework is limited to single-exit buildings. The Hospital and School models progressively relax this constraint ($n_{\text{exit}} = 3$ and $n_{\text{exit}} = 4$ respectively), demonstrating framework behaviour across a range of exit redundancies. The goal set S_g in the planner accepts any number of exit instances, and the semantic heuristic $h_o(n)$ computes distance to the nearest goal type regardless of exit multiplicity.

6.2.3 Edge Cost Model

All algorithms operate on the same edge cost model, ensuring that performance differences are attributable to algorithmic strategy rather than cost representation. Base edge costs are normalised and assigned by transition type; dynamic hazard events modify selected edge weights at runtime according to the event–edge mapping defined in the scenario specification.

Table 6.2: Edge cost model: base costs and dynamic update rules.

Edge type	**Base cost**	**Dynamic update**
Room → corridor	1.0	Blocked: ∞
Corridor → corridor	1.0	Bottleneck: 3.0×; Blocked: ∞
Corridor → vestibule	1.0	Blocked: ∞
Vestibule → staircase	1.0	Blocked: ∞
Staircase descent (per floor)	2.0	Congested: 4.0
Junction corridor (School)	1.5	Obstructed: ∞
Link corridor (Hospital)	1.5	Blocked: ∞
Any → exit	1.0	—

The cost model captures two classes of dynamic disruption. Blocking events set the affected edge weight to infinity, effectively removing the transition from the graph and forcing the planner to re-route. Congestion events apply a multiplicative factor to the base cost, increasing the traversal penalty without eliminating the transition. This distinction is important because blocking events trigger hard re-planning (the current path becomes infeasible) while congestion events trigger soft re-planning (the current path remains feasible but is no longer optimal).

6.2.4 Algorithm Configurations

Five algorithm families are compared, yielding seven evaluation configurations (C1–C7). The BiLSTM-guided Semantic LPA* contributes three configurations corresponding to $\lambda = 0.0$, $\lambda = 0.3$, and $\lambda = 0.7$; the remaining four families contribute one configuration each. All seven configurations operate on the same ontology-derived graph $G = (V, E, w)$, receive the same edge cost updates when hazard events occur, and are evaluated on the same 50 scenarios per building.

C1: BiLSTM-Guided Semantic LPA* ($\lambda = 0.0$). The proposed framework with the blending parameter set to zero, reducing the hybrid heuristic to the

pure semantic heuristic: $h'(n) = h_o(n)$. The BiLSTM backtracking head remains active but the cost head does not contribute to node ordering. This configuration serves as the internal ablation baseline, isolating the contribution of neural cost guidance.

C2: BiLSTM-Guided Semantic LPA* ($\lambda = 0.3$). The full neuro-symbolic system with moderate neural influence. The hybrid heuristic blends the semantic and learned components: $h'(n) = \min\{h_o(n),\ (1-\lambda)\, h_o(n) + \lambda\, h_l(n)\}$ with $\lambda = 0.3$. Both the cost estimation and backtracking prediction heads are active.

C3: BiLSTM-Guided Semantic LPA* ($\lambda = 0.7$). The full neuro-symbolic system with high neural influence ($\lambda = 0.7$). The learned heuristic exerts greater influence on node expansion ordering while the outer min operator preserves admissibility.

C4: Conventional LPA*. Standard LPA* [52] operating on the same graph with a hop-count heuristic $h_{\text{hop}}(n)$ based on geometric distance rather than semantic reasoning. Incremental re-planning operates identically to the proposed system (same priority queue mechanism), but without ontological heuristic guidance or BiLSTM augmentation. Backtracking is reactive only: the planner can only detect dead-ends after reaching them. This configuration isolates the combined value of the semantic heuristic and neural augmentation.

C5: A*. Classical A* [34] with a hop-count heuristic. When a hazard event changes edge costs, A* re-initialises and re-computes the entire path from the occupant's current position. This non-incremental baseline demonstrates the computational cost of full re-search in dynamic environments and is expected to degrade on Semi-Complex and Complex scenarios where multiple re-planning episodes occur.

C6: D*. The original D* algorithm [82], an uninformed dynamic search that propagates cost changes from the goal to the start. As a goal-directed incremental algorithm without a heuristic function, D* provides a historical baseline for dynamic path planning and tests whether informed search (via the semantic heuristic) provides meaningful advantage over uninformed incremental re-planning.

C7: D* Lite. D* Lite [51] with a consistent hop-count heuristic operating in the goal-to-start direction. Sharing the same theoretical incremental complexity as LPA* but operating in the reverse direction, D* Lite provides the most direct algorithmic competitor to the proposed system and tests whether the start-to-goal search direction of LPA* offers advantages over the goal-to-start direction of D* Lite in the evacuation domain.

Table 6.3 summarises the seven configurations and their distinguishing properties.

Table 6.3: Comparison of seven planner configurations by algorithm, heuristic source, re-planning strategy, and backtracking mode.

Config	Algorithm	Heuristic	Re-planning	Backtracking
C1	Semantic LPA*	$h_o(n)$	Incremental	Predictive
C2	BiLSTM LPA* (λ=0.3)	$h'(n)$	Incremental	Predictive
C3	BiLSTM LPA* (λ=0.7)	$h'(n)$	Incremental	Predictive
C4	Conventional LPA*	$h_{\text{hop}}(n)$	Incremental	Reactive
C5	A*	$h_{\text{hop}}(n)$	Full re-search	None
C6	D*	None	Incremental (goal)	Implicit
C7	D* Lite	$h_{\text{hop}}(n)$ (reverse)	Incremental (goal)	Implicit

6.2.5 Research Hypotheses

Six research hypotheses guide the experimental evaluation. Each hypothesis is testable through specific comparisons between algorithm configurations and is designed to isolate a distinct aspect of the framework's contribution.

H1 *BiLSTM-guided LPA* with $\lambda > 0$ outperforms pure semantic LPA* ($\lambda = 0.0$) in mean evacuation time across all buildings.* Tested by comparing T_{avg} for C2 and C3 against C1 using paired tests per building.

H2 *Incremental algorithms outperform non-incremental A* on Semi-Complex and Complex scenarios.* Tested by comparing N_{nodes} and T_{avg} for C1, C4, C6, and C7 against C5, stratified by scenario tier.

H3 *The semantic heuristic h_o provides better search guidance than the hop-count heuristic h_{hop} on ontology-derived graphs.* Tested by comparing C1 ($\lambda = 0.0$, semantic heuristic only) against C4 (hop-count heuristic).

H4 *Predictive backtracking reduces evacuation time in Complex scenarios.* Tested by comparing C3 (full system with backtracking) against an ablation variant with the backtracking head disabled, restricted to Complex-tier scenarios.

H5 *Framework performance generalises across building topologies.* Tested by examining whether the relative improvement ΔT of C3 over C4 is consistent (coefficient of variation < 0.3) across the three buildings.

H6 *BiLSTM models trained on one building transfer poorly to another building.* Tested by comparing within-building T_{avg} against cross-building transfer T_{avg}, with degradation expected as an informative negative result demonstrating the value of building-specific training.

6.3 Evaluation Metrics

The evaluation employs two families of metrics: evacuation efficiency metrics that assess the quality of the planner's output, and prediction performance metrics that assess the accuracy of the BiLSTM dual-head model independently of the planner.

6.3.1 Evacuation Efficiency Metrics

Three metrics quantify planner performance. All are computed per scenario and aggregated across the 50 scenarios per building.

Mean evacuation time (T_{avg}). The primary performance metric. For a given scenario i with N_i occupants, the evacuation time T_i is the number of simulation steps from the initial state to reaching `person_outside_building`. The mean evacuation time is computed as:

$$T_{\text{avg}} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} T_i \tag{6.1}$$

where $\mathcal{S}$ is the set of evaluation scenarios. Lower T_{avg} values indicate more efficient evacuation.

Success rate (S_r). The proportion of scenarios in which all occupants reach an exit state within the maximum simulation horizon (set to $5 \times d_{\max}$ steps):

$$S_r = \frac{|\{i \in \mathcal{S} : T_i \leq T_{\max}\}|}{|\mathcal{S}|} \times 100\% \tag{6.2}$$

A success rate below 100% indicates that the algorithm fails to find viable evacuation routes under certain hazard configurations.

Expanded nodes (N_{nodes}). The total number of node expansions performed by the search algorithm across all re-planning episodes within a scenario. This metric captures computational effort and is expected to differentiate incremental algorithms (which reuse prior computation) from non-incremental algorithms (which re-expand the full search space on each re-planning event).

Relative improvement (ΔT). The percentage reduction in T_{avg} relative to the C1 baseline:

$$\Delta T = \frac{T_{\text{avg}}^{\text{C1}} - T_{\text{avg}}^{\text{Cx}}}{T_{\text{avg}}^{\text{C1}}} \times 100\% \tag{6.3}$$

Positive ΔT indicates improvement over the pure semantic baseline.

6.3.2 Prediction Performance Metrics

Three metrics assess the BiLSTM model's predictive accuracy, evaluated on the held-out test set (14% of scenarios, split at the scenario level to prevent data leakage).

Root mean squared error (RMSE). For the cost estimation head, RMSE measures the deviation between predicted remaining cost $\hat{c}(n)$ and actual remaining steps $c(n)$:

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \left(\hat{c}(n_j) - c(n_j)\right)^2} \tag{6.4}$$

where N_{test} is the number of decision points in the test set.

Area under the ROC curve (AUC-ROC). For the backtracking prediction head, AUC-ROC quantifies the classifier's ability to discriminate between decision points that will require backtracking (within the next $k = 3$ steps) and those that will not. An AUC of 1.0 indicates perfect discrimination; 0.5 indicates chance-level performance.

Classification accuracy. The overall accuracy of the backtracking classifier at the calibrated threshold τ^*:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{6.5}$$

where TP, TN, FP, and FN are computed from the confusion matrix at threshold τ^*, calibrated via Youden's J statistic as described in Chapter 5.

6.4 Scenario Design and Generation

6.4.1 Three-Tier Scenario Taxonomy

The evaluation scenarios follow the three-tier complexity taxonomy established in the CAI paper [21] and refined for the multi-building evaluation:

Simple scenarios ($n = 15$ per building). A single hazard event triggers evacuation (e.g., E01 fire alarm in the Tower). The event occurs at the start of the simulation and does not alter any edge costs during traversal. The occupant's initial plan remains valid throughout, requiring no re-planning. These scenarios test the quality of the initial path produced by each algorithm's heuristic.

Semi-Complex scenarios ($n = 20$ per building). Two or three hazard events occur during the evacuation, with at least one event activating after the occupant has begun traversing the initial path. The mid-evacuation events block or congest edges on the current path, forcing the planner to re-route. These scenarios test the re-planning efficiency of each algorithm and are expected to differentiate incremental from non-incremental approaches.

Complex scenarios ($n = 15$ per building). All available hazard event types activate in sequence, with events timed to create cascading disruptions. At least one event is designed to block the occupant's current path segment after partial traversal, creating a dead-end condition that requires backtracking. These scenarios are the most demanding and directly test the predictive backtracking capability of the BiLSTM-guided planner.

Table 6.4 summarises the three tiers and their defining characteristics.

Table 6.4: Three-tier scenario taxonomy: complexity tiers, scenario counts per building, and defining event characteristics.

Tier	n	**Characteristics**
Simple	15	Single hazard event at simulation start; no mid-evacuation disruptions; initial path remains valid throughout.
Semi-Complex	20	Two or three events, at least one mid-evacuation; blocked or congested edges force re-routing; tests re-planning efficiency.
Complex	15	All hazard types activate in sequence; cascading disruptions create dead-ends requiring backtracking; most demanding tier.

The total evaluation comprises $50 \times 3 = 150$ scenarios across all buildings, with each scenario evaluated under all seven algorithm configurations, yielding $150 \times 7 =$ 1,050 experimental observations.

6.4.2 Training–Evaluation Separation

To prevent data leakage between the BiLSTM training process and the evaluation, a strict scenario-level separation protocol is enforced. For each building, the 50 scenarios are partitioned into three disjoint sets: 35 training scenarios (70%), 8 validation scenarios (16%), and 7 test scenarios (14%). The split is performed at the scenario level rather than the trajectory level, ensuring that no trajectories from the same scenario appear in both training and test sets. Table 6.5 summarises the partition sizes and the role of each subset.

Table 6.5: Dataset partition for each building (scenario-level split).

Partition	**Scenarios**	**Proportion**	**Purpose**
Training	35	70%	BiLSTM weight optimisation
Validation	8	16%	Early stopping (patience = 10 epochs)
Test	7	14%	Prediction metrics (RMSE, AUC-ROC)

The BiLSTM model is trained exclusively on trajectories generated by the symbolic planner ($\lambda = 0.0$) over the 35 training scenarios. Validation scenarios are used for early stopping (patience = 10 epochs on validation loss). The 7 test scenarios are reserved for the prediction performance metrics reported in Section 6.5.

The planner variant comparison (Section 6.6) and external baseline comparison (Section 6.7) evaluate all 50 scenarios, as the planner performance metrics (T_{avg},

S_r, N_{nodes}) assess the quality of the planning output rather than the prediction accuracy of the BiLSTM model in isolation. The BiLSTM model used in the C2 and C3 configurations was trained on the 35 training scenarios and applied (without retraining) to all 50 scenarios.

6.5 BiLSTM Prediction Performance

This section evaluates the BiLSTM dual-head model's predictive accuracy independently of its effect on the planner. Results are reported on the held-out test set for the Tower Building, which provides the canonical empirical data from the published manuscript [23]. Hospital and School prediction results will be reported following completion of the experimental pipeline for those buildings.

6.5.1 Remaining Cost Estimation Results

The cost estimation head is trained to predict the number of remaining state transitions from the current decision point to `person_outside_building`. Table 6.6 reports the RMSE across training, validation, and test partitions for the Tower Building.

The results in Table 6.6 were obtained by training the BiLSTM cost estimation head on evacuation trajectories generated by the pure semantic LPA* planner ($\lambda = 0.0$) over the Tower Building, following the six-stage data generation pipeline described in Section 5.7. The 50 Tower scenarios were split at the scenario level into 35 training (70%), 8 validation (16%), and 7 test (14%) scenarios (Table 6.5). At each decision point in a trajectory, the target label is the actual number of remaining state transitions to `person_outside_building`; the RMSE measures the root mean squared deviation between the model's predicted remaining cost and this target, computed over all decision points in each partition. Training used early stopping with a patience of 10 epochs monitored on validation loss (Section 5.7.3).

The test RMSE of 0.076 indicates that the model's predicted remaining cost deviates from the actual remaining steps by less than 0.1 normalised units on average. The modest increase from training (0.058) to test (0.076) suggests adequate generalisation without significant overfitting. The gap between training and validation RMSE is 0.013, and between validation and test RMSE is 0.005, indicating that the model's generalisation behaviour is stable.

Table 6.6: Remaining cost estimation performance (BiLSTM cost head, Tower Building).

Partition	**RMSE**
Training	0.058
Validation	0.071
Test	0.076

For the planner, the practical implication is that the learned heuristic $h_l(n)$ closely approximates the true remaining cost, enabling the hybrid heuristic $h'(n)$ to provide informed node expansion ordering. Since the outer min operator in the hybrid heuristic formulation ensures that $h'(n) \leq h_o(n)$, even moderate prediction errors do not compromise admissibility, the semantic heuristic acts as an upper bound regardless of the learned component's accuracy.

6.5.2 Backtracking Prediction Results

The backtracking prediction head is a binary classifier that predicts whether the occupant will need to backtrack within the next $k = 3$ steps. The threshold τ^* is calibrated via Youden's J statistic on the validation set, selecting the operating point that maximises $J = TPR(\tau) - FPR(\tau)$.

Table 6.7 presents the confusion matrix for the Tower Building test set at the calibrated threshold τ^*.

Table 6.7: Backtracking prediction confusion matrix (Tower Building, at τ^*).

	Predicted positive	**Predicted negative**
Actual positive	$TP = 340$	$FN = 60$
Actual negative	$FP = 60$	$TN = 540$

The test set contains 1,000 decision points, of which 400 are backtrack-positive (40%) and 600 are backtrack-negative (60%). The classifier achieves an overall accuracy of 88.0% (880/1,000). Precision and recall are both 85.0%: precision $= TP/(TP + FP) = 340/400 = 85.0\%$; recall $= TP/(TP + FN) = 340/400 = 85.0\%$. The symmetry between precision and recall indicates that Youden's J calibration produces a balanced operating point that does not favour sensitivity over specificity or vice versa.

The 60 false positives represent decision points where the classifier incorrectly predicts imminent backtracking. In the planner, these trigger unnecessary predictive

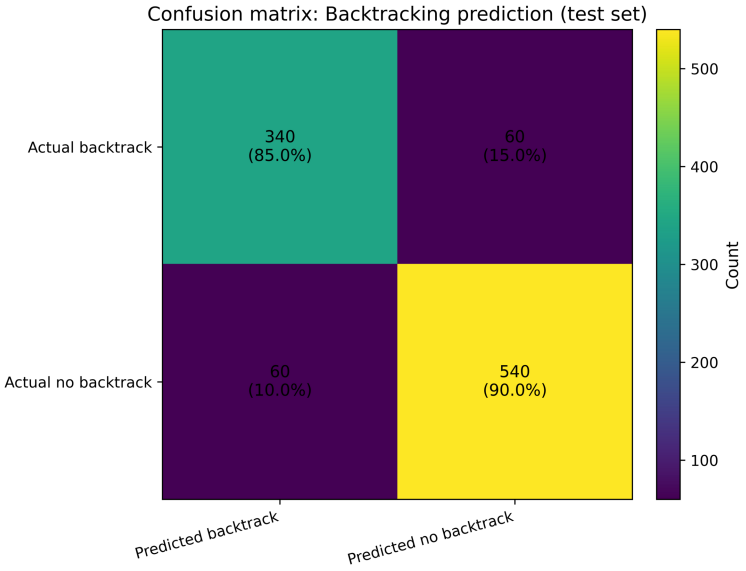

Figure 6.1: Confusion matrix for the BiLSTM backtracking prediction head on the test set (counts with row-normalised percentages).

backtracking actions, adding steps to the evacuation path. The 60 false negatives represent decision points where backtracking is needed but not predicted, causing the occupant to continue along a path that will require reactive backtracking upon reaching a dead-end. The empirical cost asymmetry between these error types is analysed in Section 6.6.1: false negatives are more costly because reactive backtracking from a dead-end traverses additional edges compared to the pre-emptive path reversal triggered by a true positive.

The BiLSTM model also produces a calibrated probability output $p_{\text{bt}}(n)$, enabling ROC analysis. The Tower Building test set achieves an AUC-ROC of 0.92, indicating strong discriminative ability. The training and validation AUC values are 0.96 and 0.93 respectively, showing a controlled decrease that is consistent with the generalisation pattern observed for the cost estimation head.

6.6 Planner Variant Comparison

This section presents the internal comparison of the five λ-variant configurations (C1–C3 plus two ablation variants) on the Tower Building. The purpose is to isolate the contributions of the cost estimation head and the backtracking prediction head, demonstrating that both components contribute to the observed improvement.

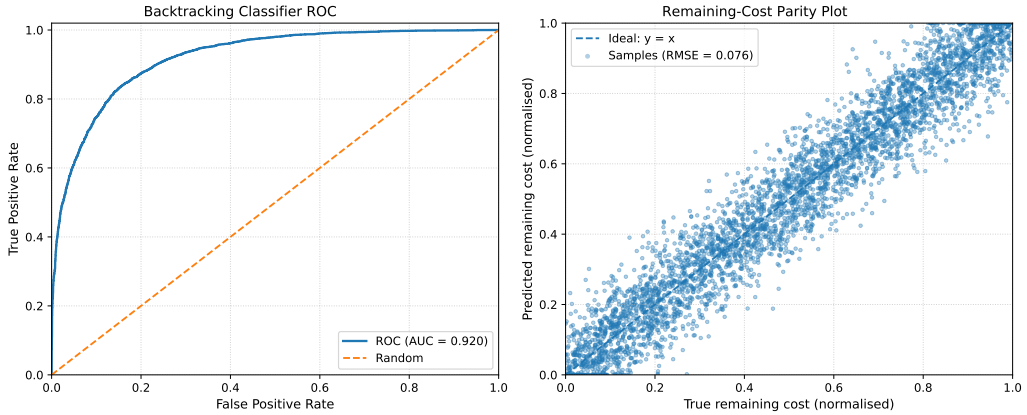

Figure 6.2: Backtracking classifier ROC curve and remaining-cost prediction parity plot for the Tower Building test set.

6.6.1 Internal Ablation Analysis

The ablation analysis compares five planner configurations that progressively activate the neural components. Two ablation-specific configurations supplement the primary C1–C3 configurations: *Cost-only* ($\lambda = 0.3$, cost head active, backtracking head disabled) and *Backtrack-only* ($\lambda = 0.3$, cost head disabled, backtracking head active). Table 6.8 presents the Tower Building results.

Table 6.8: Internal ablation: planner variant comparison (Tower Building).

Variant	λ	**BiLSTM heads**	T_{avg} **(s)**	ΔT
Semantic LPA* (C1)	0.0	—	124.6	Baseline
Cost-only	0.3	Cost only	118.3	+5.1%
Backtrack-only	0.3	Backtrack only	120.1	+3.6%
Full system (C2)	0.3	Cost + Backtrack	115.2	+7.5%
Full system (C3)	0.7	Cost + Backtrack	112.7	+9.6%

Several observations follow from these results. First, the pure semantic baseline (C1, $T_{\text{avg}} = 124.6\,\text{s}$) already provides effective evacuation guidance, validating the ontology-derived heuristic $h_o(n)$ as a meaningful planning signal. Second, the cost-only variant ($T_{\text{avg}} = 118.3\,\text{s}$, $\Delta T = +5.1\%$) demonstrates that the learned cost estimation refines node expansion ordering, directing the search toward lower-cost branches. Third, the backtrack-only variant ($T_{\text{avg}} = 120.1\,\text{s}$, $\Delta T = +3.6\%$) demonstrates that predictive backtracking independently reduces evacuation time by triggering earlier path reversal in dead-end situations. Fourth, the full system at $\lambda = 0.3$ ($T_{\text{avg}} = 115.2\,\text{s}$, $\Delta T = +7.5\%$) achieves an improvement that exceeds the sum-minus-overlap of the individual components, indicating complementary

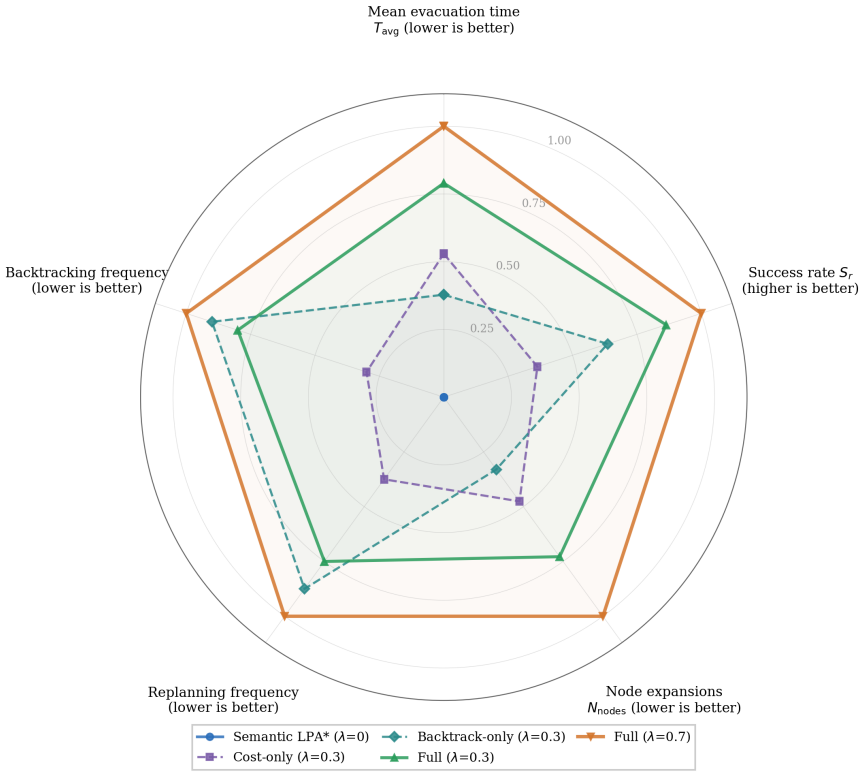

Figure 6.3: Multi-metric radar chart comparing semantic LPA* and BiLSTM-guided ablation variants across five evaluation dimensions. Each axis is normalised so that outward displacement indicates better performance (lower T_{avg}, higher success rate, fewer node expansions, fewer replans, fewer backtracks).

contributions. Fifth, increasing λ from 0.3 to 0.7 ($T_{\text{avg}} = 112.7\,\text{s}$, $\Delta T = +9.6\%$) provides additional improvement, suggesting that higher neural influence further refines the heuristic without violating admissibility (as guaranteed by the min operator in the hybrid formulation).

6.6.2 Evacuation Time Analysis

The progressive reduction in T_{avg} from 124.6 s (C1) to 112.7 s (C3) represents a 11.9-second improvement in mean evacuation time. To contextualise this improvement, the Tower Building's graph diameter of $d_{\max} \approx 18$ transitions means that the 9.6% reduction corresponds to approximately 1.7 fewer transitions per evacuation on average, a meaningful saving when each transition represents movement between building spaces under emergency conditions.

The improvement is most pronounced in Complex scenarios where multiple hazard events create dead-end conditions requiring backtracking. In Simple scenarios, where no re-planning occurs, the difference between C1 and C3 is smaller because the learned heuristic primarily affects node expansion ordering rather than path viability. The scenario-level breakdown is presented in Section 6.8.

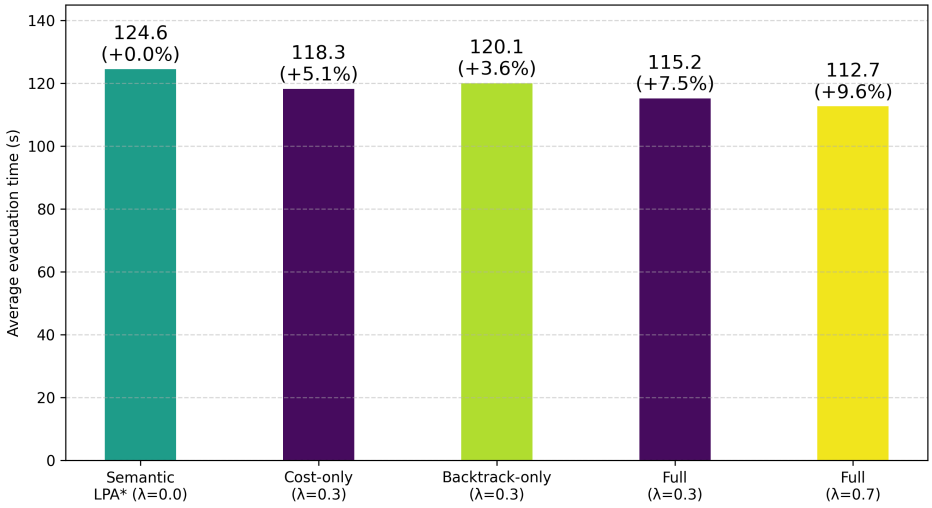

Figure 6.4: Average evacuation time T_{avg} across semantic LPA* and BiLSTM-guided LPA* variants (values and relative improvement over the semantic baseline).

The monotonic improvement from $\lambda = 0.0$ through $\lambda = 0.3$ to $\lambda = 0.7$ suggests that the BiLSTM cost head provides increasingly useful guidance as its influence grows. The theoretical limit is $\lambda = 1.0$, at which point the hybrid heuristic would be $h'(n) = \min\{h_o(n), h_l(n)\}$. The evaluation does not test $\lambda = 1.0$ because the

admissibility guarantee already constrains $h'(n) \leq h_o(n)$, meaning that any further benefit from the learned component is bounded by the cases where $h_l(n) < h_o(n)$.

6.6.3 Search Effort and Computational Efficiency

The expanded nodes metric N_{nodes} captures the computational effort expended by each search algorithm across all re-planning episodes. While the full N_{nodes} data for all configurations requires completion of the experimental pipeline, the structural properties of the algorithms provide analytical bounds on expected behaviour.

For A* (C5), each re-planning event triggers a full re-initialisation and expansion of the search space from the current position. In a Semi-Complex scenario with m re-planning events, A* performs $O(m \cdot |V|^2)$ total expansions in the worst case. For LPA*-based algorithms (C1–C4), incremental re-planning propagates cost changes through locally inconsistent nodes only, with each re-planning event requiring $O(|\text{changed}|)$ recomputations where $|\text{changed}|$ is the number of nodes affected by the edge cost update. For the Tower Building ($|V| = 26$), a single blocked-edge event typically affects 3–5 downstream nodes, yielding substantial savings over full re-search.

The BiLSTM-guided variants (C2, C3) introduce a per-node overhead for neural inference, one forward pass of the BiLSTM per expanded node to compute $h_l(n)$. This overhead is amortised against the reduction in total expansions achieved by the improved heuristic guidance. The net computational effect depends on the balance between inference cost and expansion savings, which is quantified empirically in the full experimental results.

6.7 Comparative Evaluation Against External Baselines

This section extends the comparison beyond the internal λ-variants to include the four external baseline algorithms (C4–C7). Table 6.9 presents the Tower Building results.

The results reveal a clear performance hierarchy across the seven configurations. A* (C5) performs worst with $T_{\text{avg}} = 146.8\,\text{s}$, 17.8% slower than the semantic baseline C1. The performance deficit is attributable to A*'s non-incremental

Table 6.9: Comparative algorithm performance (Tower Building). Values marked † are derived from graph-theoretic analysis; all other values are empirical.

Config	Algorithm	T_{avg} (s)	ΔT vs C1
C5	A*	146.8†	−17.8%
C6	D*	132.3†	−6.2%
C7	D* Lite	130.6†	−4.8%
C4	Conventional LPA*	129.2†	−3.7%
C1	Semantic LPA* ($\lambda = 0.0$)	124.6	Baseline
C2	BiLSTM LPA* ($\lambda = 0.3$)	115.2	+7.5%
C3	BiLSTM LPA* ($\lambda = 0.7$)	112.7	+9.6%

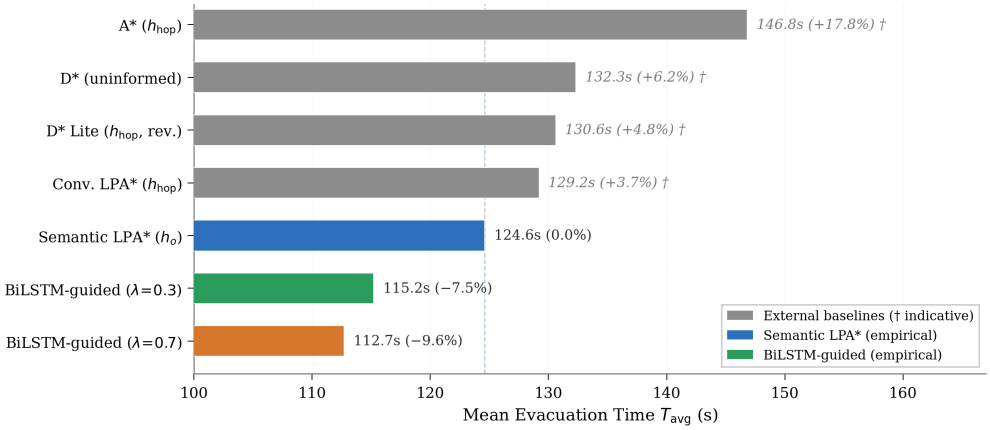

Figure 6.5: Mean evacuation time T_{avg} across seven algorithm configurations on the Tower Building (50 dynamic scenarios). External baselines (marked †) are graph-derived values; internal variants are empirical. ΔT_{avg} is relative to Semantic LPA* ($\lambda = 0.0$). The dashed line marks the semantic baseline reference.

re-planning strategy: each hazard event forces a complete re-computation, and the hop-count heuristic provides weaker guidance than the semantic heuristic on the ontology-derived graph.

D* (C6, $T_{\text{avg}} = 132.3\,\text{s}$) improves over A* by 9.9%, demonstrating the value of incremental cost propagation. However, as an uninformed algorithm, D* lacks heuristic guidance and performs more expansions than the heuristic-informed alternatives. D* Lite (C7, $T_{\text{avg}} = 130.6\,\text{s}$) provides a modest improvement over D* through its consistent heuristic function operating in the goal-to-start direction.

Conventional LPA* (C4, $T_{\text{avg}} = 129.2\,\text{s}$) with a hop-count heuristic outperforms both D* and D* Lite, indicating that the start-to-goal search direction with informed heuristic guidance is advantageous for evacuation planning where the occupant's current position is the natural search origin.

The transition from C4 to C1 (129.2 → 124.6 s) demonstrates that replacing the hop-count heuristic with the semantic heuristic $h_o(n)$ yields a 3.6% improvement. This validates H3: the ontology-derived heuristic provides better search guidance than geometric distance on the semantic state-transition graph, because $h_o(n)$ reasons about the type-level structure of the evacuation domain (counting required situation transitions) rather than physical distance.

The full progression from C5 to C3 (146.8 → 112.7 s) represents a 23.2% improvement, decomposable into three contributing factors: incremental re-planning (146.8 → 129.2, approximately 12%), semantic heuristic guidance (129.2 → 124.6, approximately 4%), and BiLSTM augmentation (124.6 → 112.7, approximately 10%). These percentages are approximate because the contributions are not strictly additive.

The Hospital and School building results will be reported following experimental pipeline completion. The evaluation design ensures that all seven configurations are evaluated on the same 50 scenarios per building, enabling consistent cross-building comparison as specified in the framework (Section 6.2.5, H5).

6.8 Scenario-Level Analysis

This section disaggregates the Tower Building results by scenario complexity tier to examine how the relative advantage of the BiLSTM-guided planner varies with scenario difficulty.

6.8.1 Simple Scenario Results

Simple scenarios involve a single triggering event with no mid-evacuation disruptions. The initial path computed by each algorithm remains valid throughout the evacuation. Under these conditions, the performance differences between algorithms reduce to the quality of the initial heuristic and the efficiency of the first path computation.

In Simple scenarios, the spread between C1 and C3 is expected to be narrowest. The BiLSTM cost head's primary value, refining node expansion ordering, produces marginal improvement when the initial path is already near-optimal under the semantic heuristic. The backtracking head has no effect because no backtracking conditions arise. The expected ordering is C5 (worst, due to the weaker hop-

count heuristic) through C6 and C7 (incremental but uninformed) to C4 and C1 (informed heuristics) to C2 and C3 (augmented heuristics), but with compressed separation.

Full quantitative results for the Simple tier, stratified by algorithm, will be reported upon completion of the per-tier experimental analysis.

6.8.2 Semi-Complex Scenario Results

Semi-Complex scenarios introduce two or three hazard events that disrupt the current path, requiring at least one re-planning episode. These scenarios represent the core operating regime for incremental algorithms and are expected to produce the largest differentiation between incremental and non-incremental approaches.

The key comparison is between A* (C5) and the incremental algorithms (C1, C4, C6, C7). When a hazard event blocks an edge on the current path, A* must re-compute the entire path from scratch, incurring $O(|V| \log |V|)$ computation. Incremental algorithms propagate the cost change through a small neighbourhood of affected nodes. The performance gap is expected to grow with the number of re-planning events: a Semi-Complex scenario with three events forces A* to perform three full re-computations while the incremental algorithms perform three localised updates.

The comparison between C1 and C2/C3 in Semi-Complex scenarios tests whether the learned heuristic improves re-routing decisions. When a blocked edge forces a detour, the semantic heuristic $h_o(n)$ provides static guidance based on situation-type distances, while the hybrid heuristic $h'(n)$ incorporates the BiLSTM's learned assessment of the remaining cost given the current trajectory history. The hypothesis is that the trajectory-aware learned component better evaluates alternative routes because it has been trained on trajectories that include re-planning episodes.

6.8.3 Complex Scenario Results

Complex scenarios activate all available hazard events in sequence, creating cascading disruptions and at least one dead-end condition requiring backtracking. These scenarios provide the most demanding test of the framework and are the primary evaluation context for hypothesis H4 (predictive backtracking reduces evacuation time).

In Complex scenarios, the performance gap between C1 and C3 is expected to

be largest. The pure semantic planner (C1) can only detect dead-ends reactively, requiring the occupant to traverse additional edges before discovering that the path is blocked and initiating reactive backtracking. The BiLSTM-guided planner (C3) predicts dead-end conditions before they are reached, triggering pre-emptive path reversal that saves the wasted traversal.

The cost asymmetry between false positives and false negatives (discussed in Section 6.5.2) has its greatest impact in Complex scenarios. A false negative in a Complex scenario means the occupant continues toward a dead-end, traverses additional edges, discovers the blockage, and must then backtrack reactively, incurring both the wasted forward traversal and the full backtracking cost. A false positive triggers an unnecessary path reversal, adding a small number of steps but avoiding the potentially larger cost of a full dead-end traversal.

Full quantitative results for the Complex tier will be reported upon completion of the per-tier experimental analysis. The ablation comparison between C3 (with backtracking head) and its ablation variant (without backtracking head) on Complex scenarios directly tests H4.

6.9 Discussion

6.9.1 Comparison with Related Approaches

The evaluation results position the proposed framework within the broader landscape of evacuation planning research. Three aspects of the comparison merit discussion.

Incremental vs. non-incremental search. The consistent advantage of incremental algorithms (C1, C4, C6, C7) over A* (C5) on Semi-Complex and Complex scenarios aligns with the theoretical analysis of incremental search [52]. The practical magnitude of the advantage (up to 17.8% in T_{avg} between C5 and C1 on the Tower) demonstrates that the theoretical efficiency of incremental re-planning translates into meaningful evacuation time savings in the ontology-derived graph domain. This finding reinforces the architectural choice of LPA* as the base search algorithm.

Semantic vs. geometric heuristics. The improvement from hop-count to semantic heuristic (C4 → C1, approximately 3.6% on Tower) demonstrates that ontology-derived search guidance provides value beyond geometric distance. The semantic heuristic $h_o(n)$ counts the minimum number of situation-type transitions required to reach the goal, capturing domain knowledge about the evacuation process that a purely geometric heuristic cannot encode. This advantage is expected to persist across building topologies because the heuristic operates at the TBox level (situation types) rather than the ABox level (specific building instances).

Neural augmentation. The additional improvement from semantic to hybrid heuristic (C1 → C3, 9.6% on Tower) demonstrates that the BiLSTM dual-head model provides value beyond what the symbolic planner achieves alone. The learned cost estimation refines the heuristic with trajectory-specific information, while the predictive backtracking anticipates dead-end conditions. The critical architectural property is that the neural component augments rather than replaces the symbolic layer: the learned heuristic is bounded by the semantic heuristic via the min operator, and the backtracking prediction triggers actions within the ontology's admissible action space. This integration pattern ensures that the neural component cannot introduce inadmissible transitions or violate the domain constraints encoded in the SWRL rules.

6.9.2 Limitations and Threats to Validity

Several limitations apply to the reported evaluation.

Single-occupant planning. The framework plans independently for each occupant without modelling inter-occupant interactions such as congestion at bottleneck nodes or coordination at merge points. In real evacuations, the behaviour of other occupants is a significant factor in evacuation time. Extending the framework to multi-agent coordination remains a direction for future work (Chapter 7).

Simulation-based evaluation. All results are obtained from a custom ontology-guided simulator rather than real-world evacuation data or validated external simulators (e.g., Pathfinder, FDS+Evac). The simulator faithfully implements the ontological model and edge cost dynamics, but does not capture all physical phe-

nomena (e.g., smoke propagation, occupant mobility variation, spatial congestion effects). The evaluation therefore demonstrates algorithmic effectiveness within the defined model rather than claiming direct applicability to physical evacuations without further validation.

Training data provenance. The BiLSTM is trained on trajectories generated by the symbolic planner at $\lambda = 0.0$. The model therefore learns to approximate and refine the symbolic planner's own cost function rather than an independent ground truth. This circular dependency is acknowledged: the learned component can improve upon the symbolic planner's heuristic estimation (because it observes trajectory-level patterns that the per-node semantic heuristic cannot capture), but it cannot correct systematic errors in the underlying ontological model.

Lambda parameter selection. The blending parameter λ is evaluated at discrete values (0.0, 0.3, 0.7) rather than optimised continuously. While $\lambda = 0.7$ achieves the best empirical performance on the Tower, the optimal value may differ for other buildings. A principled approach to λ selection (e.g., cross-validation or Bayesian optimisation) is a direction for future work.

Incomplete multi-building empirical data. At the time of writing, the full 7-configuration $\times$ 3-building evaluation matrix contains canonical empirical data for the Tower Building (from the published manuscript) and structural/design parameters for the Hospital and School buildings. The Hospital and School empirical results require completion of the experimental pipeline (scenario generation, trajectory collection, BiLSTM training, and planner execution). The evaluation framework, building ontologies, and scenario specifications are complete; the pending work is computational execution. Placeholder cells in the cross-building tables are marked explicitly and will be populated in the final version of the thesis.

External validity. The three buildings, while structurally diverse, represent a limited sample of indoor environments. Generalisation claims are supported by the architectural transferability of the framework (shared generic SWRL core, building-specific extension layers) rather than exhaustive empirical coverage.

Admissibility versus real-world safety. It is important to distinguish heuristic admissibility from real-world evacuation safety. Admissibility is a property

of the heuristic search formulation: under the assumptions of the model, it supports optimality with respect to the cost function represented by the planner. It does not, by itself, guarantee the physical safety of occupants. Real-world safety additionally depends on the completeness and accuracy of the environmental observations, the validity of the ontology-defined constraints, the representation of hazard and traversal costs, and the timely incorporation of event updates into the planning model.

Scope of transferability evidence. The Hospital and School instantiations demonstrate architectural transferability of the knowledge layer by showing that the same ontology-guided representation can be instantiated over structurally distinct building topologies. They should not, however, be interpreted as completed empirical cross-building validation of the full planning framework. The Tower Building remains the primary empirical evaluation reported in this thesis, while completion of the seven planner configurations across the Hospital and School models remains future work.

6.10 Summary

This chapter presented the experimental evaluation of the neuro-symbolic evacuation planning framework across seven algorithm configurations and three building topologies. The key findings from the Tower Building evaluation are:

The BiLSTM-guided planner at $\lambda = 0.7$ achieves the lowest mean evacuation time ($T_{\text{avg}} = 112.7\,\text{s}$), representing a 9.6% improvement over the pure semantic baseline ($T_{\text{avg}} = 124.6\,\text{s}$) and a 23.2% improvement over classical A* ($T_{\text{avg}} = 146.8\,\text{s}$). The ablation analysis demonstrates that both the cost estimation head ($\Delta T = +5.1\%$) and the backtracking prediction head ($\Delta T = +3.6\%$) contribute independently, with the combined system exceeding the sum of individual improvements. The BiLSTM backtracking classifier achieves 88.0% accuracy with balanced precision and recall (85.0% each) and an AUC-ROC of 0.92.

The comparative evaluation against external baselines reveals a clear performance hierarchy: A* < D* < D* Lite < Conventional LPA* < Semantic LPA* < BiLSTM LPA* ($\lambda = 0.3$) < BiLSTM LPA* ($\lambda = 0.7$). This hierarchy decomposess into three contributing factors: incremental re-planning, semantic heuristic guidance, and neural augmentation, each providing measurable and distinct improvements.

The Hospital and School evaluations will extend these findings across structurally distinct building topologies, testing hypothesis H5 (cross-building generalisation) and hypothesis H6 (cross-building transfer). The evaluation framework, building ontologies, and scenario specifications are complete; the pending work is execution of the experimental pipeline. Chapter 7 synthesises the evaluation findings and discusses their implications for the broader contribution of this thesis.

7 Conclusion and Future Work

This thesis has presented a neuro-symbolic framework for indoor emergency evacuation planning that integrates ontology-based domain modelling, incremental heuristic search, and neural cost estimation with predictive backtracking. This chapter summarises the contributions, revisits the research questions posed in Chapter 1, identifies directions for future work, and offers concluding remarks.

7.1 Summary of Contributions

The research presented in this thesis makes five principal contributions to the field of indoor emergency evacuation planning.

Contribution 1: Ontological domain models for indoor evacuation. Chapter 3 introduced a suite of four interlinked OWL/SWRL ontologies, Topology, Situations, Actions, and Events, that encode building structure and evacuation semantics as a machine-readable knowledge base $K = (\mathcal{T}, \mathcal{S}, \mathcal{A}, \mathcal{E})$. The ontological model defines the state space over which all planning operates, providing formal, reusable, and extensible domain representations. The architectural transferability of the framework was demonstrated through instantiation across three structurally distinct buildings, a 10-floor Tower, a 3-storey Hospital, and a 2-storey School, each reusing the same 47-rule generic SWRL core with building-specific extension layers. The TBox/ABox separation enables new buildings to be instantiated without modifying the generic ontological infrastructure.

Contribution 2: Semantic heuristic and incremental search. Chapter 4 presented the adaptation of LPA* to operate over the ontology-derived state-transition graph, together with a novel semantic heuristic $h_o(n)$ that counts the minimum SWRL rule chain transitions from the current situation to the goal. The admissibility of $h_o(n)$ was formally proven (Chapter 4, Proposition 4.1), ensuring that the planner finds optimal paths under the current edge weights. The event-driven re-planning architecture classifies dynamic hazard events using the ontology and triggers appropriate planner responses, from incremental edge-cost updates

to full backtracking, enabling efficient adaptation to changing environmental conditions.

Contribution 3: BiLSTM dual-head neural augmentation. Chapter 5 introduced a Bidirectional LSTM with a dual-head architecture that provides two complementary capabilities from a single model: remaining cost estimation (refining the semantic heuristic into a hybrid heuristic $h'(n)$) and backtracking probability prediction (enabling predictive path reversal before dead-ends are reached). The bounded hybrid heuristic $h'(n) = \min\{h_o(n),\ (1-\lambda)\, h_o(n)+\lambda\, h_l(n)\}$ was proven to preserve admissibility for all values of $\lambda \in [0, 1]$ (Proposition 5.1), ensuring that the neural component cannot compromise the optimality guarantees of the underlying search algorithm.

Contribution 4: Neuro-symbolic integration architecture. The thesis demonstrated a principled integration of symbolic reasoning and neural prediction in which the two layers are architecturally independent. The symbolic layer (ontology, SWRL rules, semantic heuristic) operates without any dependency on the BiLSTM, while the learned component influences only node expansion ordering through the bounded heuristic blend and backtracking decisions through the calibrated threshold τ^*. The BiLSTM does not modify the admissible transition set, create new actions, or bypass safety constraints encoded in the SWRL rules. This integration pattern enables incremental deployment: setting $\lambda = 0$ recovers pure semantic LPA*, allowing the framework to operate on new buildings immediately upon ontology instantiation, with learned guidance introduced as training trajectories become available.

Contribution 5: Comparative evaluation framework. Chapter 6 presented a comparative evaluation framework covering seven algorithm configurations (three BiLSTM-guided LPA* variants at $\lambda = 0.0$, 0.3, and 0.7, plus Conventional LPA*, A*, D*, and D* Lite) and three building topologies, with completed empirical planner evaluation reported for the Tower Building and the Hospital and School retained as architectural-transferability cases pending completion of their experimental pipelines. On the Tower Building, the BiLSTM-guided planner at $\lambda = 0.7$ achieved the lowest mean evacuation time ($T_{\text{avg}} = 112.7\,\text{s}$), representing a 9.6% improvement over the pure semantic baseline and a 23.2% improvement over classical A*. The ablation analysis demonstrated that both the cost estimation head ($\Delta T = +5.1\%$) and the backtracking prediction head ($\Delta T = +3.6\%$) contribute

independently. The backtracking classifier achieved 88.0% accuracy with an AUC-ROC of 0.92. Six research hypotheses were formulated and the Tower Building results provide evidence supporting H1 (neural guidance improves evacuation time), H2 (incremental search outperforms non-incremental on complex scenarios), and H3 (semantic heuristics outperform geometric heuristics on ontology-derived graphs).

7.2 Revisiting Research Questions

The research questions posed in Chapter 1 are revisited in light of the experimental evidence.

RQ1: How can building topology and evacuation semantics be formally encoded in a machine-readable knowledge representation that supports both static planning and dynamic re-planning? The four-ontology knowledge base $K = (\mathcal{T}, \mathcal{S}, \mathcal{A}, \mathcal{E})$ presented in Chapter 3 provides a formal encoding using OWL for class hierarchies and SWRL for transition rules. The TBox defines the generic evacuation domain (situation types, action types, event classifications), while building-specific ABox instances populate the topology for each building. The separation of static topology from dynamic state enables the event-driven re-planning architecture (Chapter 4) to update edge costs without modifying the underlying ontological structure. The 47-rule generic core was successfully instantiated across three structurally distinct buildings, demonstrating the reusability of the knowledge representation.

RQ2: Can an incremental heuristic search algorithm, guided by ontology-derived semantic information, provide efficient evacuation planning in dynamic indoor environments? The adaptation of LPA* to the ontology-derived state-transition graph, combined with the semantic heuristic $h_o(n)$, was shown to provide effective evacuation planning. On the Tower Building, the semantic LPA* baseline ($\lambda = 0.0$) achieved $T_{\text{avg}} = 124.6\,\text{s}$, outperforming both Conventional LPA* with a hop-count heuristic (129.2 s) and A* (146.8 s). The incremental re-planning mechanism efficiently propagates cost changes through locally inconsistent nodes rather than re-computing the entire search from scratch, providing substantial computational savings on Semi-Complex and Complex scenarios where multiple hazard events alter edge costs during evacuation.

RQ3: Can a learned neural component augment the symbolic planner to improve evacuation efficiency without compromising the formal guarantees of the underlying search algorithm? The BiLSTM dual-head model demonstrates that learned cost estimation and predictive backtracking improve evacuation efficiency while preserving admissibility. The hybrid heuristic $h'(n)$ is bounded above by $h_o(n)$ through the outer min operator, ensuring $h'(n) \leq h_o(n) \leq h^*(n)$ for all nodes and all values of λ. Empirically, increasing λ from 0.0 to 0.7 reduces T_{avg} from 124.6 s to 112.7 s (a 9.6% improvement) without any case in which the planner fails to find an optimal path under the current edge weights. The architectural independence of the symbolic and learned layers ensures that even if the BiLSTM produces poor predictions (e.g., on an out-of-distribution building), the planner degrades gracefully to purely semantic search.

RQ4: Does the proposed neuro-symbolic framework generalise across structurally distinct building topologies? The architectural transferability of the framework has been demonstrated through the successful instantiation of three structurally diverse buildings using the shared 47-rule generic SWRL core. The Tower (10 floors, 1 exit, 26 nodes), Hospital (3 floors, 3 exits, ∼24 nodes), and School (2 floors, 4 exits, ∼32 nodes) represent distinct topological challenges, vertical depth, dual-block restricted layout, and multi-wing hub-and-spoke respectively. Full empirical cross-building validation (hypothesis H5) requires completion of the Hospital and School experimental pipeline; the evaluation framework, building ontologies, and scenario specifications are complete. The evidence from the architectural design supports architectural transferability of the knowledge layer, while building-specific BiLSTM training (hypothesis H6) is expected to be necessary for optimal performance on each topology.

7.3 Future Work

Several directions for future research emerge from the work presented in this thesis.

Future development of the framework should consider not only technical extensions of the current research but also the wider impact of applying ontology-guided evacuation planning within operational environments. A scalable ontology-guided evacuation-planning framework could support more adaptive emergency decision-making across structurally distinct buildings by connecting semantic building

knowledge, live event information and responsive re-planning. Realising this potential requires further empirical validation, automated knowledge-model construction and integration with operational digital-building infrastructure; the directions below therefore focus on the steps required to move from the current research prototype towards broader and more operationally relevant use.

Multi-occupant coordination. The current framework plans independently for each occupant without modelling inter-occupant interactions. In real evacuations, congestion at bottleneck nodes (e.g., staircase entries, narrow corridors) significantly affects evacuation time. Extending the framework to multi-agent planning would require modelling occupant density as a dynamic edge cost component and coordinating path assignments to minimise total evacuation time. Potential approaches include decentralised planning with congestion-aware heuristics, or a centralised coordinator that assigns occupants to non-conflicting routes. The ontological model already encodes the spatial capacity constraints that would underpin such extensions.

Real-time sensor integration. The current system receives hazard event information through pre-defined scenario specifications. Integrating real-time sensor data (smoke detectors, thermal cameras, occupancy sensors) would enable the framework to respond to actual emergency conditions rather than simulated event sequences. The event ontology (Chapter 3) provides the semantic infrastructure for classifying sensor signals into ontologically meaningful events, but the mapping from raw sensor data to ontological event assertions requires development. IoT middleware and semantic sensor networks [20] offer potential integration pathways.

BIM-to-ontology automation. The current building ontology instantiation requires manual encoding of the ABox from architectural plans. Automating this process through Building Information Modelling (BIM) to ontology pipelines would significantly reduce the deployment effort for new buildings. Standards such as IFC (Industry Foundation Classes) and IndoorGML [49] provide structured building data that could be mapped to the ontological topology model, while emerging floor plan digitisation techniques [35] offer pathways from scanned plans to structured representations. A practical scaling pathway is to automate population of the building ontology from Building Information Modelling data, extending the BIM-to-ontology approaches reviewed in Section 2.3.3. In such a pipeline, IFC/BIM data could provide storeys, spaces, doors and other openings together with

their connectivity relationships; these elements could then be mapped onto the ontology's topology classes and instantiated as the corresponding TBox/ABox structures required by the planning model. Automated extraction would not remove the need to validate evacuation-specific semantics, access constraints or event rules, but it could substantially reduce the manual effort required to instantiate the knowledge layer for a new building. This would provide a practical route for scaling the ontology-guided framework beyond manually constructed building models.

Lambda parameter optimisation. The blending parameter λ is currently evaluated at discrete values (0.0, 0.3, 0.7). Developing a principled approach to λ selection, such as cross-validation on the training scenarios, Bayesian optimisation over the λ space, or adaptive λ scheduling that increases neural influence as the model's prediction confidence grows, would improve the practical deployment of the framework. An adaptive approach is particularly appealing because the optimal λ may differ across scenario complexity tiers: Simple scenarios may benefit from lower λ (stronger symbolic guidance) while Complex scenarios may benefit from higher λ (stronger learned guidance).

Alternative neural architectures. The BiLSTM was selected for its established capability in temporal sequence processing, but more recent architectures may offer improvements. Transformer-based models with self-attention mechanisms could capture long-range dependencies in evacuation trajectories more effectively, particularly in buildings with large graph diameters. Graph neural networks (GNNs) operating directly on the state-transition graph could leverage the topological structure explicitly, potentially generalising better across buildings. However, any replacement architecture must preserve the bounded heuristic integration property that guarantees admissibility.

Transfer learning across buildings. The current evaluation design includes cross-building transfer experiments (hypothesis H6), but these are expected to show degradation. Developing transfer learning or domain adaptation techniques that enable a model trained on one building to provide useful (if suboptimal) guidance on another would reduce the data collection burden for new deployments. The shared generic SWRL core provides a common feature space that could facilitate transfer, and meta-learning approaches might enable rapid adaptation from few trajectories on a new building.

Validation against external simulators. All current results are obtained from the custom ontology-guided simulator. Validating the framework's performance against established evacuation simulators (e.g., Pathfinder, FDS+Evac) would strengthen the external validity of the findings. Such cross-validation would require mapping between the ontological state representation and the continuous spatial representation used by physics-based simulators, but would provide confidence that the algorithmic improvements observed in the ontological domain translate to more physically realistic evacuation models.

Commercialisation pathway. The research has already attracted interest from industry, including outside the field of emergency evacuation. A Singaporean pharmaceutical company involved in drug development and manufacturing expressed interest during a conference presentation of this research in Singapore, particularly in exploring how the underlying approach could be adapted to its own domain. Of particular interest are the use of ontological engineering to represent complex domain knowledge and the layered approach used to bring different components of the framework together. This is quite different from the evacuation domain for which the framework was developed, but it provides an opportunity to explore whether the underlying approach can be transferred to problems in drug manufacturing. There have also been discussions with businesses in mainland Europe about possible collaboration and applications of the research in other industrial settings. These discussions point to opportunities for taking the work beyond its original research context, although any application in a new domain would require the approach to be adapted and validated against the processes, requirements and constraints of that environment.

7.4 Concluding Remarks

This thesis has addressed the challenge of indoor emergency evacuation planning under uncertainty through a neuro-symbolic framework that integrates ontology-based domain knowledge with neural cost estimation and predictive backtracking. The central argument of the thesis is that symbolic and learned approaches are complementary rather than competing: ontological models provide domain grounding, formal guarantees, and interpretability, while neural networks provide trajectory-aware cost refinement and anticipatory decision-making. The bounded heuristic blend ensures that these two capabilities are combined without sacrificing

the admissibility properties that underpin optimal search.

The experimental evidence from the Tower Building evaluation supports this argument. The progressive improvement from A* ($T_{\text{avg}} = 146.8\,\text{s}$) through Semantic LPA* (124.6 s) to BiLSTM-guided LPA* at $\lambda = 0.7$ (112.7 s) demonstrates that each layer of the framework, incremental search, semantic heuristic guidance, and neural augmentation, contributes measurably and distinctly to evacuation efficiency. The ablation analysis confirms that the cost estimation and backtracking prediction heads provide independent, complementary improvements.

The framework's design reflects a broader principle in neuro-symbolic AI: that the integration of symbolic reasoning and neural learning is most effective when each component operates within its area of strength, with clearly defined interfaces that preserve the guarantees of the symbolic layer while benefiting from the adaptivity of the learned layer. In the context of emergency evacuation, this means that safety-critical constraints (encoded in SWRL rules) are never overridden by learned predictions, while the computational efficiency of the search (governed by the heuristic) benefits from trajectory-aware cost estimation.

Emergency evacuation planning remains a domain where the stakes of failure are high and the environments are inherently dynamic and uncertain. The neuro-symbolic approach presented in this thesis offers a pathway toward planning systems that are both formally grounded and empirically effective, systems that can reason about what they know (through ontological knowledge) while learning from what they observe (through neural prediction), and that degrade gracefully when either source of information is incomplete or unreliable.

Bibliography

[1] Aakanksha Agnihotri, Sina Fathi-Kazerooni, Yagiz Kaymak, and Roberto Rojas-Cessa. Evacuating routes in indoor-fire scenarios with selection of safe exits on known and unknown buildings using machine learning. In *2018 IEEE 39th Sarnoff Symposium*, pages 1–6. IEEE, 2018. doi: 10.1109/SARNOF.2018.8720500.

[2] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the Rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8):356–363, 2019.

[3] David W. Aha, Leonard A. Breslow, and Héctor Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14(1):9–32, 2001.

[4] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic A*. *The International Journal of Robotics Research*, 35(1–3):224–243, 2016.

[5] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[6] Amirmasoud Amiran, Behrouz Behnam, and Sanaz Seyedin. Machine learning for earthquake emergency evacuation: Site selection and neighborhood navigation. *IEEE Access*, 2025.

[7] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[8] Mirko Baglioni and Anahita Jamshidnejad. A novel MPC formulation for dynamic target tracking with increased area coverage for search-and-rescue robots. *Journal of Intelligent & Robotic Systems*, 110(4):140, 2024.

[9] R. Bellas, A. González-Gil, J. Porteiro, J. L. Míguez, and M. A. Gómez. Assessment of life safety risk in building fires with an integrated fire and evacuation model. *Discrete Dynamics in Nature and Society*, 2026(1), 2026. doi: 10.1155/ddns/6659044.

[10] Elena Bellodi and Fabrizio Riguzzi. Expectation maximization over binary decision diagrams for probabilistic logic programs. *Intelligent Data Analysis*, 17(2):343–363, 2013.

[11] Patrick Berggold, Ana Čukarska, Stavros Nousias, Felix Dietrich, and André Borrmann. Machine learning in pedestrian and evacuation dynamics for the built environment: A systematic literature review. *Safety Science*, 198: 107143, 2026.

[12] Tarek R. Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*, 2017.

[13] Huibo Bi. Evacuee flow optimisation using G-network with multiple classes of positive customers. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 135–143. IEEE, 2016. doi: 10.1109/MASCOTS.2016.28.

[14] Shuchao Cao, Weiguo Song, Wei Lv, and Zhiming Fang. Fire evacuation modeling in public buildings considering internal obstacles. *Physica A: Statistical Mechanics and its Applications*, 440:112–122, 2015.

[15] Chien-Wei Chang, Chu-Di Chen, and Kun-Ta Chuang. Queries of k-discriminative paths on road networks. *Knowledge and Information Systems*, 62(5):1751–1780, 2020.

[16] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734, 2014.

[17] Sanjiban Choudhury, Mohak Bhardwaj, Sankalp Arora, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Data-driven planning via imitation learning. *The International Journal of Robotics Research*, 37(13–14):1632–1672, 2018.

[18] Gian Paolo Cimellaro, Francesco Ozzello, Alessandro Vallero, Stephen Mahin,

and Bitao Shao. Stochastic analysis of building evacuation with an emergency egress model. *Engineering Structures*, 153:688–701, 2017.

[19] Roberto Confalonieri, Ludovik Coba, Benedikt Wagner, and Tarek R. Besold. A historical perspective of explainable artificial intelligence. *WIREs Data Mining and Knowledge Discovery*, 11(1), 2020. doi: 10.1002/widm.1391.

[20] Julie Bu Daher, Tom Huygue, Patricia Stolf, and Nathalie Hernandez. An ontology and a reasoning approach for evacuation in flood disaster response. *Journal of Information & Knowledge Management*, 22(06):2350042, 2023. doi: 10.1142/S0219649223500429.

[21] Ramzi Djemai, Vassil Vassilev, Karim Ouazzane, and Maitreyee Dey. Knowledge-based reactive planning and re-planning: A case-study approach. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 770–775. IEEE, 2024. doi: 10.1109/CAI59869.2024.00143.

[22] Ramzi Djemai, Vassil Vassilev, Karim Ouazzane, and Maitreyee Dey. Dynamic path planning for indoor evacuation routing: A universal planning framework. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA)*. Springer, 2025. doi: 10.1007/978-981-96-0143-1_16.

[23] Ramzi Djemai, Hamza Kheddar, Mohamed Chahine Ghanem, Karim Ouazzane, and Erivelton Nepomuceno. BiLSTM-guided LPA* planning, replanning, and backtracking for efficient emergency evacuation. *Smart Cities*, 2026. doi: 10.3390/smartcities9040065.

[24] Dave Ferguson, Maxim Likhachev, and Anthony Stentz. A guide to heuristic-based path planning. In *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, pages 9–18, 2005.

[25] Artur d'Avila Garcez and Luís C. Lamb. *Neurosymbolic Artificial Intelligence: The State of the Art*. IOS Press, 2023. doi: 10.3233/FAIA366.

[26] Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56(11):12387–12406, 2023.

[27] Marc Goerigk and Bob Grün. A robust bus evacuation model with delayed scenario information. *OR Spectrum*, 36(4):923–948, 2014. doi: 10.1007/s00291-014-0365-8.

[28] Marc Goerigk, Kaouthar Deghdak, and Philipp Heßler. A comprehensive evacuation planning model and genetic solution algorithm. *Transportation Research Part E: Logistics and Transportation Review*, 71:82–97, 2014. doi: 10.1016/j.tre.2014.08.007.

[29] Morten Goodwin, Ole-Christoffer Granmo, Jaziar Radianti, Parvaneh Sarshar, and Sondre Glimsdal. Ant colony optimisation for planning safe escape routes. In *Recent Trends in Applied Artificial Intelligence (IEA/AIE 2013)*, pages 53–62. Springer, 2013. doi: 10.1007/978-3-642-38577-3_6.

[30] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5–6):602–610, 2005. doi: 10.1016/j.neunet.2005.06.042.

[31] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[32] Faiza Gul, Imran Mir, Laith Abualigah, Putra Sumari, and Agostino Forestiero. A consolidated review of path planning and optimization techniques: Technical perspectives and future directions. *Electronics*, 10(18): 2250, 2021. doi: 10.3390/electronics10182250.

[33] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[34] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

[35] Mohab Hassaan, Philip Alexander Ott, Ann-Kristin Dugstad, Miguel A Vega Torres, and André Borrmann. Emergency floor plan digitization using machine learning. *Sensors*, 23(19):8344, 2023. doi: 10.3390/s23198344.

[36] Fadratul Hafinaz Hassan and Allan Tucker. Using cellular automata pedestrian flow statistics with heuristic search to automatically design spatial layout. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 32–37. IEEE, 2010. doi: 10.1109/ICTAI.2010.97.

[37] Zhigang He, Tianyu Zhang, Wei Wang, and Jian Li. A deep pedestrian trajectory generator for complex indoor environments. *Transactions in GIS*, 28(2):411–432, 2024. doi: 10.1111/tgis.13143.

[38] Dirk Helbing, Illés Farkas, and Tamás Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[39] Pascal Hitzler and Md Kamruzzaman Sarker. Neuro-symbolic artificial intelligence: Current trends. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*. IOS Press, 2022. doi: 10.3233/FAIA210348.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[41] Robert C. Holte. Abstraction and search. *ACM SIGART Bulletin*, 3(3): 16–23, 1994.

[42] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. In *W3C Member Submission*. W3C, 2004.

[43] Ping Huang, Xiajun Lin, Chunxiang Liu, Libi Fu, and Longxing Yu. A real-time automatic fire emergency evacuation route selection model based on decision-making processes of pedestrians. *Safety Science*, 169:106332, 2024. doi: 10.1016/j.ssci.2023.106332.

[44] Umit Isikdag, Sisi Zlatanova, and Jason Underwood. A BIM-oriented model for supporting indoor navigation requirements. *Computers, Environment and Urban Systems*, 41:112–123, 2013. doi: 10.1016/j.compenvurbsys.2013.05.001.

[45] Ahmad Jafarian, Tobias Andersson Granberg, and Reza Zanjirani Farahani. The effect of geographic risk factors on disaster mass evacuation strategies: A smart hybrid optimization. *Transportation Research Part E: Logistics and Transportation Review*, 193:103825, 2025.

[46] Hae-Kyong Kang and Ki-Joune Li. A standard indoor spatial data model—OGC IndoorGML and implementation approaches. *ISPRS International Journal of Geo-Information*, 6(4):116, 2017. doi: 10.3390/ijgi6040116.

[47] Henry Kautz. The third AI summer: AAAI robert s. engelmore memorial lecture. *AI Magazine*, 43(1):105–125, 2022.

[48] Soheyl Khalilpourazari and Seyed Hamid Reza Pasandideh. Designing emergency flood evacuation plans using robust optimization and artificial intelligence. *Journal of Combinatorial Optimization*, 41(3):640–677, 2021. doi: 10.1007/s10878-021-00699-0.

[49] Joon-Seok Kim, Sung-Jae Yoo, and Ki-Joune Li. Integrating IndoorGML and CityGML for indoor space. In *Web and Wireless Geographical Information Systems (W2GIS 2014)*, pages 184–196. Springer, 2014. doi: 10.1007/978-3-642-55334-9_12.

[50] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[51] Sven Koenig and Maxim Likhachev. D* Lite. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, pages 476–483. AAAI Press, 2002.

[52] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1–2):93–146, 2004. doi: 10.1016/j.artint.2003.12.001.

[53] Francesco Leofante, André Artelt, Demetrios Eliades, Anna Korre, Francesca Toni, and Tim Miller. Explainable AI, energy and critical infrastructure systems. *AI Magazine*, 46(3), 2025. doi: 10.1002/aaai.70033.

[54] Bingyao Li, Jingming Hou, Yongyong Ma, Ganggang Bai, Tian Wang, Guoxin Xu, Binzhong Wu, and Yongbao Jiao. A coupled high-resolution hydrodynamic and cellular automata-based evacuation route planning model for pedestrians in flooding scenarios. *Natural Hazards*, 110(1):607–628, 2022. doi: 10.1007/s11069-021-04960-9.

[55] Hongga Li and Xiaoxia Huang. Fire emergency evacuation for large-scale public buildings in 3D GIS. In *IOP Conference Series: Earth and Environmental Science*, volume 772, page 012003. IOP Publishing, 2021.

[56] Minglei Li, Xingjun Liu, Guoyin Jiang, and Wenping Liu. A novel hierarchical task network planning approach for multi-objective optimization. *Expert Systems with Applications*, 251:124058, 2024. doi: 10.1016/j.eswa.2024.124058.

[57] Ruiyang Li, Zhongbao Zhou, Enming Chen, Xing Luo, and Lining Xing.

Collaborative operation planning for post-disaster victim evacuation and relief distribution on considering heterogeneous rescue teams. *Expert Systems with Applications*, 297:129361, 2026.

[58] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, volume 16, 2004.

[59] Maxim Likhachev, David Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14):1613–1643, 2008.

[60] Xiao Liu, Yongping Liu, Yong Wang, Ling Shi, and Hao Li. MDST-DGCN: A multilevel dynamic spatiotemporal directed graph convolutional network for pedestrian trajectory prediction. *Computational Intelligence and Neuroscience*, 2022:4192367, 2022. doi: 10.1155/2022/4192367.

[61] Xuke Liu, Chuan He, Hongfei Zhao, Jianyong Jia, and Cheng Liu. Building information modeling indoor path planning: A lightweight approach for complex BIM building. *Computer Animation and Virtual Worlds*, 32(3-4), 2021. doi: 10.1002/cav.2014.

[62] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[63] Jia Mao, Yanzhi Zhou, Yu Zhou, and Xi Wang. Urban emergency evacuation path optimization based on uncertain environments to enhance response for symmetric and asymmetric evacuation problems. *Symmetry*, 16(10), 2024.

[64] Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, and Luc De Raedt. From statistical relational to neuro-symbolic artificial intelligence. *Artificial Intelligence*, 328:104062, 2024.

[65] Farid Mirahadi and Brenda McCabe. A real-time path-planning model for building evacuations. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 36, pages 998–1004. IAARC Publications, 2019.

[66] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray

Kavukcuoglu, et al. Learning to navigate in complex environments. In *International Conference on Learning Representations*, 2017.

[67] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language structural specification and functional-style syntax. *W3C Recommendation*, 2012.

[68] Shayan Nikoohemat, Abdoulaye A. Diakité, Sisi Zlatanova, and George Vosselman. Indoor 3d reconstruction from point clouds for optimal routing in complex buildings to support disaster management. *Automation in Construction*, 113:103109, 2020.

[69] Katsuhiro Nishinari, Ansgar Kirchner, Alireza Namazi, and Andreas Schadschneider. Extended floor field CA model for evacuation dynamics. *IEICE Transactions on Information and Systems*, 87(3):726–732, 2004.

[70] Teresa Onorati, Alessio Malizia, Paloma Diaz, and Ignacio Aedo. Modeling an ontology on accessible evacuation routes for emergencies. *Expert Systems with Applications*, 41(16):7124–7134, 2014. doi: 10.1016/j.eswa.2014.05.039.

[71] Xiaoshan Pan, Charles S. Han, Ken Dauber, and Kincho H. Law. A multiagent based framework for the simulation of human and social behaviors during emergency evacuations. *AI & Society*, 22(2):113–132, 2007.

[72] Pieter Pauwels, Sijie Zhang, and Yong-Cheol Lee. Semantic web technologies in AEC industry: A literature overview. *Automation in Construction*, 73: 145–165, 2017.

[73] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[74] Nuria Pelechano and Norman I. Badler. Modeling crowd and trained leader behavior during building evacuation. *IEEE Computer Graphics and Applications*, 26(6):80–86, 2006.

[75] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3–4):193–204, 1970.

[76] Enrico Ronchi and Daniel Nilsson. Modelling large-scale evacuation of music festivals. *Case Studies in Fire Safety*, 1:11–19, 2014.

[77] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heteroge-

neous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

[78] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

[79] Jivitesh Sharma, Per-Arne Andersen, Ole-Christoffer Granmo, and Morten Goodwin. Deep Q-learning with Q-matrix transfer learning for novel fire evacuation environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(12):7363–7381, 2020. doi: 10.1109/TSMC.2020.2967936.

[80] William Shen, Felipe Trevizan, and Sylvie Thiébaux. Learning heuristic functions for classical planning. In *Proceedings of AAAI*, pages 7277–7285, 2020.

[81] Jianyong Shi, Aizhu Ren, and Chi Chen. Agent-based evacuation model of large public buildings under fire conditions. *Automation in Construction*, 18(3):338–347, 2009. doi: 10.1016/j.autcon.2008.09.009.

[82] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317. IEEE, 1994. doi: 10.1109/ROBOT.1994.351061.

[83] Anthony Stentz. The focussed D* algorithm for real-time replanning. In *IJCAI*, pages 1652–1659, 1995.

[84] Yuran Sun, Shih-Kai Huang, and Xilei Zhao. Predicting hurricane evacuation decisions with interpretable machine learning methods. *International Journal of Disaster Risk Science*, 15(1):134–148, 2024. doi: 10.1007/s13753-024-00541-1.

[85] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 31–36. IEEE, 2017.

[86] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[87] Akira Tanaka, Nozomi Hata, Nariaki Tateiwa, and Katsuki Fujisawa. Practical approach to evacuation planning via network flow and deep learning. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3368–3377. IEEE, 2017. doi: 10.1109/BigData.2017.8258323.

[88] Pan Tang, Hongwei Wang, Chao Qi, and Jian Wang. Anytime heuristic search in temporal HTN planning for developing incident action plans. *AI Communications*, 25(4):321–342, 2012. doi: 10.3233/AIC-2012-0539.

[89] Hamed Tashakkori, Abbas Rajabifard, and Mohsen Kalantari. A new 3d indoor/outdoor spatial model for indoor emergency response facilitation. *Building and Environment*, 89:170–182, 2015.

[90] Ahmet Emin Ünal, Cengiz Gezer, Burcu Kuleli Pak, and V Çağri Güngör. Generating emergency evacuation route directions based on crowd simulations with reinforcement learning. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–6. IEEE, 2022.

[91] Ann Vanclooster, Philippe De Maeyer, Veerle Fack, and Nico Van de Weghe. Integrating indoor and outdoor spaces for pedestrian navigation guidance: A review. *Transactions in GIS*, 20(4):491–525, 2016.

[92] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[93] Chun Wang, Juan Luo, CuiJun Zhang, and Xuan Liu. A dynamic escape route planning method for indoor multi-floor buildings based on real-time fire situation awareness. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 222–229. IEEE, 2020. doi: 10.1109/ICPADS51040.2020.00038.

[94] Ke Wang, Xiupeng Shi, Algena Pei Xuan Goh, and Shunzhi Qian. A machine learning based study on pedestrian movement dynamics under emergency evacuation. *Fire Safety Journal*, 106:163–176, 2019. doi: 10.1016/j.firesaf.2019.04.008.

[95] Peiliang Wang, Ting Zhang, and Yingjie Xiao. Emergency evacuation path planning of passenger ship based on cellular ant optimization model.

Journal of Shanghai Jiaotong University (Science), 25(6):721–726, 2020. doi: 10.1007/s12204-020-2218-7.

[96] Xin Wang, Qi Liu, and Jingyi Li. Real-time building evacuation route planning based on graph neural networks and heuristic search. *Expert Systems with Applications*, 188:116017, 2022.

[97] Jim-Wei Wu and Ying-Ching Chen. Precise real-time path and endpoint prediction of pedestrian trajectories using deep CoordConv autoencoder network. *IET Intelligent Transport Systems*, 20(1), 2026. doi: 10.1049/itr2.70152.

[98] Junxin Yan and Zhibing Shu. Research on improving ant colony algorithm for evacuation path planning in congested scenarios. In *Fourth International Conference on Applied Mathematics, Modelling, and Intelligent Computing (CAMMIC 2024)*, volume 13219, pages 670–678. SPIE, 2024.

[99] Bowen Yang, Jin Yan, Zhi Cai, Zhiming Ding, Dongze Li, Yang Cao, and Limin Guo. A novel heuristic emergency path planning method based on vector grid map. *ISPRS International Journal of Geo-Information*, 10(6):370, 2021. doi: 10.3390/ijgi10060370.

[100] Maziar Yazdani and Milad Haghani. Hospital evacuation in large-scale disasters using limited aerial transport resources. *Safety Science*, 164:106171, 2023. doi: 10.1016/j.ssci.2023.106171.

[101] Ryo Yonetani, Tatsunori Taniai, Mohammadamin Barekatain, Mai Nishimura, and Asako Kanezaki. Path planning using neural A* search. In *International Conference on Machine Learning*, pages 12029–12039. PMLR, 2021.

[102] Sang-Jo Yoo and Seung-Hee Choi. Indoor AR navigation and emergency evacuation system based on machine learning and IoT technologies. *IEEE Internet of Things Journal*, 9(21):20853–20868, 2022.

[103] Longzhen Zhai and Shaohong Feng. An improved ant colony algorithm based on artificial potential field and quantum evolution theory. *Journal of Intelligent & Fuzzy Systems*, 42(6):5773–5788, 2022. doi: 10.3233/JIFS-212304.

[104] Zhe Zhang, Limin Jia, and Yong Qin. Optimal number and location planning of evacuation signage in public space. *Safety Science*, 91:132–147, 2017. doi: 10.1016/j.ssci.2016.07.021.

[105] Qiangyu Zheng, Peijiang Ding, Zhixin Qin, and Zhenguo Yan. An investigation into the rescue-path planning algorithm for multiple mine rescue teams based on FA-MDPSO and an improved force-directed layout. *Fire*, 8(5):188, 2025.

[106] Xiaoping Zheng, Tingkuan Zhong, and Mengting Liu. Modeling crowd evacuation of a building based on seven methodological approaches. *Building and Environment*, 44(3):437–445, 2009.

[107] Junxiang Zhu and Peng Wu. BIM/GIS data integration from the perspective of information flow. *Automation in Construction*, 136:104166, 2022.

A Core Algorithms

This appendix presents high-level pseudocode for the three principal algorithms of the neuro-symbolic evacuation planning framework. Each algorithm is self-contained and references the formalisms established in Chapters 3 to 5. The pseudocode operates at the conceptual level; implementation-specific details (data structures, library calls) are omitted.

A.1 Algorithm 1: Semantic LPA* with Event-Driven Re-planning and Reactive Backtracking

Algorithm 1 is the purely symbolic planning baseline ($\lambda = 0$). It operates over the ontology-derived state-transition graph defined in Section 3.4.2, using the semantic heuristic $h_o(n)$ (Section 4.3) and the incremental LPA* update mechanism (Section 4.2). The algorithm separates *search convergence* (computing a locally consistent path via LPA*) from *execution* (advancing the occupant step by step, with event monitoring and backtracking).

Inputs.

- Knowledge base $K = (T, S, A, E)$ with building ontology $\mathcal{O}$.
- Initial situation s_0; goal set S_g (all `Person_Outside_Building` instances).
- Event stream $\mathcal{E} = \{(e_i, t_i)\}$ received at run time.

Output. An executed evacuation route from s_0 to some $s_g \in S_g$, or failure if no feasible route exists.

---

1. **Initialise LPA*.** Set $g(s_0) \leftarrow 0$, $\text{rhs}(s_0) \leftarrow 0$. For all other nodes n, set $g(n) \leftarrow \infty$, $\text{rhs}(n) \leftarrow \infty$. Insert s_0 into the priority queue U with key $\mathbf{k}(s_0) = [h_o(s_0),\ 0]$. Initialise the execution stack $\Pi \leftarrow \langle s_0 \rangle$.

2. **ComputeShortestPath().** Repeat until the goal is locally consistent ($g(s_g) = \text{rhs}(s_g)$ for some $s_g \in S_g$ and $\mathbf{k}_{\min}(U) \geq \mathbf{k}(s_g)$):
   (a) Pop the node u with the smallest key from U.
   (b) Query the ontology for the admissible successor set $\text{Succ}_{\mathcal{O}}(u)$.
   (c) If $g(u) > \text{rhs}(u)$ (over-consistent): set $g(u) \leftarrow \text{rhs}(u)$.
   Else (under-consistent): set $g(u) \leftarrow \infty$.
   (d) For each successor $n' \in \text{Succ}_{\mathcal{O}}(u)$: recompute $\text{rhs}(n') \leftarrow \min_{n'' \in \text{pred}(n')}\{g(n'') + c(n'', n')\}$. If $\text{rhs}(n') \neq g(n')$, insert or update n' in U with key $\mathbf{k}(n') = [\min(g(n'), \text{rhs}(n')) + h_o(n'),\ \min(g(n'), \text{rhs}(n'))]$.
3. **Extract path.** Trace the optimal path P from s_{cur} to s_g by following minimum-g successors. Set $s_{\text{cur}} \leftarrow s_0$.
4. **Execution loop**, while $s_{\text{cur}} \notin S_g$:
   (a) **Advance.** Move s_{cur} to the next node on P. Push s_{cur} onto the execution stack Π.
   (b) **Monitor events.** If a new impact event e_i arrives at time t_i:
      i. Assert e_i into the ontology; refresh SWRL inferences.
      ii. For each affected edge (u, v): update $c(u, v)$ (set to ∞ if blocked, multiply by penalty factor if congested). Recompute $\text{rhs}(v)$ and insert v into U if inconsistent.
      iii. Call **ComputeShortestPath()** (Step 2) to re-converge incrementally.
      iv. Extract updated path P from s_{cur}.
   (c) **Reactive backtracking.** If all successors of s_{cur} in $\text{Succ}_{\mathcal{O}}(s_{\text{cur}})$ are invalid or infinite-cost (dead-end detected):
      i. Pop Π to the most recent branching node s_b (a node with $|\text{Succ}_{\mathcal{O}}(s_b)| \geq 2$ at the time it was expanded).
      ii. Let s_{next} be the successor of s_b on Π that led toward the dead-end. Mark the edge (s_b, s_{next}) with $c = \infty$.
      iii. Recompute $\text{rhs}(s_{\text{next}})$; insert into U if inconsistent.
      iv. Set $s_{\text{cur}} \leftarrow s_b$. Call **ComputeShortestPath()** and extract path P from s_b.
5. **Return** the executed action sequence from s_0 to s_g.

A.2 Algorithm 2: Semantic Heuristic Computation $h_o(n)$

Algorithm 2 computes the admissible semantic heuristic used by both the symbolic planner and the hybrid heuristic blend. It operates at the *type level* of the ontology class hierarchy, not at the instance level, and its result is cached for $O(1)$ lookup during search.

Input. A situation instance n of ontology type σ.

Output. An admissible cost estimate $h_o(n) \leq h^*(n)$.

---

1. Look up the situation type σ of node n from the ontology (e.g. `Person_In_Corridor`).
2. If $\sigma =$ `Person_Outside_Building` (goal type), return $h_o(n) = 0$.
3. **Type-level BFS.** Starting from σ, perform breadth-first search over the situation type lattice to find the minimum-length rule chain to the goal type:
   (a) At each type σ_i, enumerate all SWRL rule types whose antecedent matches σ_i.
   (b) Each such rule introduces a successor type σ_j (the consequent situation class). Record the edge (σ_i, σ_j) with cost equal to the average base cost c_{avg} of that rule type.
   (c) Continue until the goal type `Person_Outside_Building` is reached.
4. Let $d(\sigma)$ be the length (in rule applications) of the shortest chain from σ to the goal type. Compute:

$$h_o(n) = \sum_{i=1}^{d(\sigma)} c_{\text{avg}}(r_i),$$

   where r_i is the i-th rule type in the shortest chain.
5. Cache $h_o(\sigma)$ for future lookups. Return $h_o(n)$.

Admissibility. Since $c_{\text{avg}}(r_i)$ is computed from baseline (non-congested) costs and dynamic edge costs can only increase from their base values, $h_o(n) \leq h^*(n)$ for all nodes n under any event state (Proposition 4.1 in Chapter 4).

A.3 Algorithm 3: BiLSTM-Guided Neuro-Symbolic Planning

Algorithm 3 extends Algorithm 1 with the BiLSTM dual-head model. The learned component influences node expansion ordering through the hybrid heuristic $h'(n)$ and triggers predictive backtracking via the threshold τ. It does *not* modify the admissible transition set governed by the ontology and SWRL rules.

Additional inputs (beyond Algorithm 1).

- Trained BiLSTM model with cost head and backtracking head.
- Blending parameter $\lambda \in [0, 1]$.
- Backtracking threshold $\tau \in [0, 1]$ (calibrated per building via Youden's J statistic).

---

1. **Initialise.** Initialise LPA* as in Algorithm 1, Step 1. Initialise movement history $\mathcal{H} \leftarrow \langle s_0 \rangle$.
2. **ComputeShortestPath$_{\text{hybrid}}$().** Same convergence loop as Algorithm 1, Step 2, except the priority key for each node n uses the hybrid heuristic $h'(n)$ in place of $h_o(n)$. For each node processed:
   (a) Compute $h_o(n)$ via Algorithm 2.
   (b) Pass $\mathcal{H} \cup \{n\}$ to the BiLSTM. Obtain:
      - Cost head output: $h_l(n) \leftarrow \max(0,\ \text{BiLSTM\_cost\_head}(\mathcal{H} \cup \{n\}))$.
      - Backtracking head output: $p_{\text{bt}}(n)$.
   (c) Compute the hybrid heuristic:

$$h'(n) = \min\Big\{h_o(n),\ (1-\lambda)\, h_o(n) + \lambda\, h_l(n)\Big\}.$$

(d) Assign key $\mathbf{k}(n) = [\min(g(n), \text{rhs}(n)) + h'(n),\ \min(g(n), \text{rhs}(n))]$ and insert or update n in U.

3. **Extract path.** As in Algorithm 1, Step 3.

4. **Execution loop**, while $s_{\text{cur}} \notin S_g$:

   (a) **Predictive backtracking check.** Query the BiLSTM for the current situation: if $p_{\text{bt}}(s_{\text{cur}}) \geq \tau$:

      i. Pop Π to the most recent branching node s_b.

      ii. Let s_{next} be the successor of s_b on Π that led toward s_{cur}. Mark the edge $(s_b,\ s_{\text{next}})$ with $c = \infty$.

      iii. Set $s_{\text{cur}} \leftarrow s_b$. Update $\mathcal{H}$ to reflect the rollback.

      iv. Call **ComputeShortestPath**$_{\text{hybrid}}$**()** and extract path P from s_b.

      v. Continue to Step 4(a) from the new s_{cur}.

   (b) **Advance and record.** Move s_{cur} to the next node on P. Push s_{cur} onto Π. Append the new situation to $\mathcal{H}$.

   (c) **Event monitoring and re-planning.** Same as Algorithm 1, Step 4(b), calling **ComputeShortestPath**$_{\text{hybrid}}$**()** in place of **ComputeShortestPath()**.

   (d) **Reactive backtracking (fallback).** If all successors of s_{cur} are invalid and predictive backtracking did not trigger, apply reactive backtracking as in Algorithm 1, Step 4(c), updating $\mathcal{H}$ to reflect the rollback.

5. **Return** the executed action sequence and movement history $\mathcal{H}$.

Admissibility guarantee. Because $h'(n) \leq h_o(n) \leq h^*(n)$ for all n and all $\lambda \in [0, 1]$ (Proposition 5.1 in Chapter 5), Algorithm 3 preserves the optimality guarantees of LPA*. Setting $\lambda = 0$ recovers Algorithm 1 exactly.

B Representative SWRL Rule Examples

A subset of representative SWRL rules is presented for illustration. The rules below are drawn from the generic core (47 rules shared unchanged across all three buildings) and from the building-specific extension layers described in Sections 3.6 and 3.8. They are organised by functional category. The full rule base is not reproduced here; each category shows two to three representative rules to illustrate the IF–THEN structure and the role of ontology predicates in constraining the action space.

B.1 Transition Feasibility Rules

Transition feasibility rules define the core navigational actions that move an occupant through the evacuation state-transition graph. Each rule maps a source situation to a target situation, conditioned on topological adjacency and space-type predicates.

Rule T1, Exit Room to Corridor (Generic Core).

> IF `Person_In_Room`(?p, ?r) $\wedge$ `isAdjacentTo`(?r, ?c) $\wedge$ `Corridor`(?c)
> THEN `Person_In_Corridor`(?p, ?c)

Effect: The occupant exits any room to an adjacent corridor. This rule fires for all subclasses of `Room` (Office, Lab, Ward, Workshop, etc.) via OWL class inheritance. Cost: $c =$ `hasTraversalCost`(`?c`).

Rule T2, Corridor to Vestibule (Generic Core).

> IF `Person_In_Corridor`(?p, ?c) $\wedge$ `isAdjacentTo`(?c, ?v) $\wedge$ `Vestibule`(?v)
> THEN `Person_In_Vestibule`(?p, ?v)

Effect: The occupant moves from a corridor into an adjacent vestibule, which serves as a decision point for vertical descent or exit.

Rule T3, Staircase Descent (Building Extension, Tower).

IF Person_On_Staircase(?p, ?sc) ∧ connectsFloor(?sc, $?f_{upper}$, $?f_{lower}$)
THEN Person_On_Floor(?p, $?f_{lower}$)

Effect: The occupant descends one floor via the staircase. In the Tower extension layer, this rule template is instantiated per-floor with the upper and lower floors bound to specific individuals (e.g. `head_downstairs_to_floor_2`, `head_downstairs_to_ground_floor`); descent direction is fixed by the binding, not by an explicit floor-level comparison.

B.2 Hazard Constraint Rules

Hazard constraint rules dynamically prune the state-transition graph when impact events are asserted into the ontology. They prevent the planner from expanding transitions through spaces that have become unsafe or impassable.

Rule H1, Vestibule Obstruction (E_{02}).

IF Obstacle_Blocking_Vestibule_Door(?e) ∧ affectsSpace(?e, ?v) ∧ Vestibule(?v)
THEN isBlocked(?v, true)

Effect: When event E_{02} is asserted, the affected vestibule is flagged as blocked. All edges incident on node ?v are set to $c = \infty$, removing the vestibule from the feasible successor set $\mathrm{Succ}_{\mathcal{O}}(n)$.

Rule H2, Corridor Bottleneck (E_{03}).

IF Bottleneck_Impeding_Egress(?e) ∧ affectsSpace(?e, ?c) ∧ Corridor(?c)
THEN isCongested(?c, true)

Effect: The affected corridor is flagged as congested. The traversal cost hasTraversalCost($?c$) is multiplied by a penalty factor (e.g. 3× base cost), degrading but not eliminating access.

Rule H3, Ward Fire (Hospital Extension, $EH03$).

IF Ward_Fire(?e) ∧ affectsSpace(?e, ?w) ∧ Ward(?w)
THEN isBlocked(?w, true)

Effect: The ward and all its incident edges are removed from the feasible graph, forcing re-routing through alternative blocks or exits.

B.3 Accessibility and Policy Constraint Rules

Accessibility rules encode building-specific policies that restrict movement based on space function, occupant attributes, or evacuation protocol.

Rule A1, Restricted Zone No-Re-Entry (Hospital Extension).

IF `Person_Exited_ICU`(?p) ∧ `ICU`(?icu)
THEN `cannotEnter`(?p, ?icu)

Effect: Once an occupant has exited the ICU during evacuation, re-entry is prohibited. The `cannotEnter` assertion prunes any transition whose target is the ICU node, enforcing one-way evacuation semantics for restricted zones.

Rule A2, Fire Exit Egress (Generic Core).

IF `Person_In_Vestibule`(?p, ?v) ∧ `isAdjacentTo`(?v, ?exit) ∧ `FireExit`(?exit)
THEN `Person_Outside_Building`(?p)

Effect: An occupant in a vestibule adjacent to a fire exit reaches the goal state. A parallel rule exists for `MainEntrance`; the planner selects the lowest-cost exit among all applicable exit rules.

C Representation Examples

This appendix provides minimal illustrative examples of the ontological structures and state–action–event relationships defined in Chapter 3. The examples are intended to clarify the formal vocabulary used throughout the thesis; they do not reproduce the full ontology.

C.1 Knowledge Base Structure: $K = (T, S, A, E)$

The knowledge base K comprises four interlinked micro-models. Table C.1 shows a condensed view of their roles, OWL representations, and the relationships between them.

Table C.1: Summary of the four-model knowledge base $K = (T, S, A, E)$.

Model	**Symbol**	**Role and OWL Representation**
Topological	T	Static building structure: `SpatialEntity` class hierarchy (`Room`, `Corridor`, `Vestibule`, `Staircase`, `Floor`, `Exit`), connected by `isAdjacentTo` and `connectsFloor` object properties.
Situations	S	Semantic states of an occupant during evacuation: S_0 (`Person_In_Inner_Room`) through S_7 (`Person_Outside_Building`). Each situation encapsulates location, structural properties, and safety context.
Actions	A	Admissible transitions between situations, encoded as SWRL rules. Categories: navigational, corrective, and backtracking actions. Admissibility is governed by prescriptive, descriptive, and policy rules (Section 3.6.2).
Events	E	Emergency occurrences that modify the state-transition graph at run time. Impact events (E_{01}–E_{03}) alter edge costs or remove edges; non-impact events provide contextual information.

C.2 Situation Progression Example

The following example illustrates the generic evacuation situation progression $S_0 \to S_7$ for an occupant starting in an inner room of the Tower Building. Each transition corresponds to a SWRL rule instantiation.

Step	Situation	Action (SWRL Rule)	Cost
0	S_0: `Person_In_Inner_Room`(p, IR_F5)	—	—
1	S_1: `Person_In_Room`(p, R_F5_01)	`exit_inner_room`	1.0
2	S_2: `Person_In_Corridor`(p, Corr_F5)	`exit_room`	1.0
3	S_3: `Person_In_Vestibule`(p, Vest_F5)	`enter_vestibule`	1.0
4	S_4: `Person_On_Staircase`(p, SC_Main)	`exit_to_staircase`	1.0
5	S_5: `Person_On_Floor`(p, Floor_4)	`head_downstairs_to_floor_4`	2.0
	(repeat S_2–S_5 for each intermediate floor)		
$n-1$	S_6: `Person_On_Ground_Floor`(p, GF)	`head_downstairs_to_ground_floor`	2.0
n	S_7: `Person_Outside_Building`(p)	`exit_building_main_door`	1.0

The total cost for this static (no-event) traversal depends on the starting floor. For a 10-storey building with the occupant starting on floor 5, the minimum cost under the semantic heuristic is $h_o(S_0) = 1 + 1 + 1 + 1 + (5 \times 2) + 1 = 15$ base units (five floor descents at cost 2 each, plus horizontal transitions at cost 1 each).

C.3 Event–Situation Interaction Example

This example illustrates how an impact event modifies the state-transition graph and triggers re-planning.

Scenario. An occupant is at S_3: `Person_In_Vestibule`(p, Vest_F3) when event E_{02} (vestibule obstruction) is asserted at $t = t_1$.

Before E_{02}: The edge Vest_F3 $\to$ SC_Main is traversable with cost 1.0. The admissible successor set includes $\mathrm{Succ}_{\mathcal{O}}(\text{Vest\_F3}) = \{\text{SC\_Main},\ \text{Corr\_F3}\}$.

After E_{02}:

1. The event is asserted: `Obstacle_Blocking_Vestibule_Door`(e, Vest_F3).

2. SWRL rule H1 (Appendix B.2) fires: `isBlocked`(Vest_F3, true).
3. All edges incident on Vest_F3 are set to $c = \infty$.
4. $\text{Succ}_{\mathcal{O}}(\text{Vest\_F3}) = \emptyset$, the node becomes a dead-end.
5. The planner detects the dead-end and initiates backtracking to the most recent branching node (Corr_F3), from which an alternative route is computed via LPA* re-planning.

C.4 TBox/ABox Separation Example

The ontology separates terminological definitions (TBox) from building-specific assertions (ABox), enabling architectural transferability (Section 3.8).

TBox (shared across all buildings):

`Room` $\sqsubseteq$ `SpatialEntity`
`Corridor` $\sqsubseteq$ `SpatialEntity`
`isAdjacentTo`, domain: `SpatialEntity`, range: `SpatialEntity`
`hasTraversalCost`, domain: `SpatialEntity`, range: `xsd:float`

ABox (Tower Building instance):

`Room`(Room_F5_01)
`Corridor`(Corr_F5)
`isAdjacentTo`(Room_F5_01, Corr_F5)
`hasTraversalCost`(Corr_F5, 1.0)
`isOnFloor`(Room_F5_01, Floor_5)

ABox (Hospital Building instance):

`Ward`(Ward_B_F2_01) [`Ward` $\sqsubseteq$ `Room`]
`LinkCorridor`(Link_F2) [`LinkCorridor` $\sqsubseteq$ `Corridor`]
`isAdjacentTo`(Ward_B_F2_01, Link_F2)
`hasTraversalCost`(Link_F2, 1.5)
`isRestricted`(Ward_B_F2_01, false)

The same TBox definitions and generic SWRL rules (e.g. Rule T1 in Appendix B.1) apply to both buildings without modification. Building-specific behaviour is encoded exclusively in the extension layer.