Reasoning on Geo-Referenced Sensor Data in Physical Infrastructures

Sebastian Mieth[1], Florian Fuchs[1], Edgar-Philipp Stoffel[2], and Diana Weiß[2]

[1] Siemens AG Corporate Technology, Intelligent Autonomous Systems,
sebastianmieth@gmail.com, florian.fuchs.ext@siemens.com,
Otto-Hahn-Ring 6, 81739 Munich, Germany,
[2] Ludwig-Maximilians-Universität München, Institute for Informatics,
stoffel@pms.ifi.lmu.de, diana.weiss@ifi.lmu.de,
Oettingenstraße 67, 80538 Munich, Germany

Abstract. This paper introduces a generic approach to integrating different kinds of geo-referenced sensor measurements along a physical infrastructure. The underlying core ontology is domain-independent and realized using Semantic Web technologies; it can be specialized for different domains. In particular, railway infrastructures are presented as a case study. Using the physical infrastructure as a common spatial reference system constitutes a central point of the integration, which allows to perform reasoning tasks, such as answering network-related queries (involving measurements from both stationary and mobile sensors). A classification of different query types is presented together with the corresponding algorithms.

1 Introduction

Physical infrastructures (e.g., transportation networks and energy distribution networks) are vital for the functioning of the society. Ensuring their operational reliability is challenging due to their large geographical extent, the complex interdependencies between infrastructure components, and the large number of organizations involved. Monitoring the current state of a physical infrastructure is therefore essential for both operations and maintenance.

The large extent of most physical infrastructures requires monitoring by automatically acquiring state information using sensor systems. Various such systems (both mobile and stationary) are currently deployed in physical infrastructures: railway infrastructures are, e. g., equipped with pressure sensors for wheel impact load, infrared cameras for hot axle box detection, and laser scanners for track geometry measurements; power networks use current sensors, insulating gas density sensors at circuit breakers, and voltage sag/swell sensors.

A lack of integration, however, frequently prevents comprehensive interpretation and reasoning on the available sensor measurements. A major reason is the lack of a common spatial reference system for sensor measurements, especially in the case of mobile sensor systems. Other reasons are the lack of a generic approach for integrating measurements from different sources and the lack of a

formal model of sensor semantics. This currently requires tailored solutions for each individual integration requirement.

This work describes a domain-independent approach for integrating geo-referenced sensor measurements in physical infrastructures using Semantic Web technologies. Section 2 identifies requirements and compares them with related work. Section 3 proposes a hierarchical spatial network as a common spatial reference system in combination with a generic ontology as a common sensor semantics model. The mapping of geo-referenced sensor measurements to the spatial network is described in Section 4. Section 5 shows how to utilize the network topology for geospatial reasoning on the integrated sensor measurements. An appropriate software design for implementing the proposed systems is presented in Section 6. Finally, Section 7 discusses the approach and concludes the paper.

2 Requirements and Related Work

This section elaborates on requirements of a generic, domain-independent approach for the integration of geo-referenced sensor measurements in physical infrastructures. In addition, related work is introduced and matched with these requirements.

2.1 Requirements

In order to use the physical infrastructure as a reference system for sensor measurements, a generic spatial network model needs to be designed. This model should support different kinds of domains including railways and energy distribution networks. Furthermore, it needs to be scalable and easily manageable since typical physical infrastructures are composed of a significant number of interrelated components like e.g. tracks and rail switches in the railway domain.

Also required are mechanisms for a generic integration of sensor data into the proposed spatial network model: for each infrastructure element (e.g., track or switch), the relevant sensor readings should be retrievable like the wheel impact load measurement or the hot axle box detection. In order to achieve this, sensor readings have to be associated to infrastructure elements. This is especially difficult for mobile sensors like train mounted sensors. Consequently, mechanisms need to be developed that determine the relevance of sensor data with respect to infrastructure elements.

Finally, a generic, extensible set of query types for searching within the physical infrastructure has to be supported to allow e.g. the query for an optimal freight train route taking maximum speed restrictions of single tracks into account. Here, it should be possible to support a number of domain-independent, reusable queries as well as to augment this set by new, domain-dependent ones.

2.2 Related Work

A number of approaches already deal with sensor integration in physical infrastructures:

In the field of energy distribution networks, General Electric Co. developed the Power Management Control System [1] for monitoring the electrical landscape of energy facilities. In the academic area, Salmenjoki et al. enhance electricity distribution systems by using agents and semantic web technologies [2].

Dailey et al. introduced a distributed intelligent transportation system [3] to apply to domains where real-time data is asynchronously provided by spatially distributed, heterogeneous sensor systems like road networks.

The Siemens RailCom Manager [4] combines information, communication, and control systems related to railroads. Its main purpose is to provide access to the above mentioned systems via a single, integrated interface. A train inspection system is introduced by Maly et al. [5]. For this, all relevant train properties are acquired to get knowledge of the complete train condition for better failure estimation.

All these systems focus on one particular domain, but do not propose a generic approach. Moreover, they assume an *existing* mapping of sensors to components of the physical infrastructure, so a generic integration of arbitrary types of sensors cannot be found.

3 Spatial Network Model

This section presents our underlying spatial network model of a physical infrastructure. Its purpose is twofold: First, it serves as a spatial reference system for anchoring and integrating different sensor measurements and, thereupon, as a basis for advanced query processing (discussed in Sections 5, 6). In order to meet the previously posed requirements of scalability and manageabilty, we use *hierarchical graphs*. Advantages of this approach are, in particular:

- They can be helpful for constraining search space (e.g., for finding a path between two stations in the province Bavaria, the partial networks of other provinces such as Hesse or Saxony need not be considered).
- They allow to insert intermediate abstraction levels as required due to the inherent *partitioning* of data. This partitioning also renders a decentralized administration possible, such as in a Web environment. Moreover, query answering can benefit from it, since load balancing becomes possible.

A spatial network graph consists of nodes (e.g. switches, terminals, train stations) and edges (e.g. tracks) – both are referred to as the *network elements* in the subsequent.

Centers (depicted by cubes in Fig. 1) are introduced as basic organizational units (e.g. corresponding to individual Web servers in a distributed environment) with a distinct, associated *area of responsibility* (*AoR*) : a center c_0 governs the sub-graph, G_{c_0}, of the network graph spatially contained in its AoR. We further

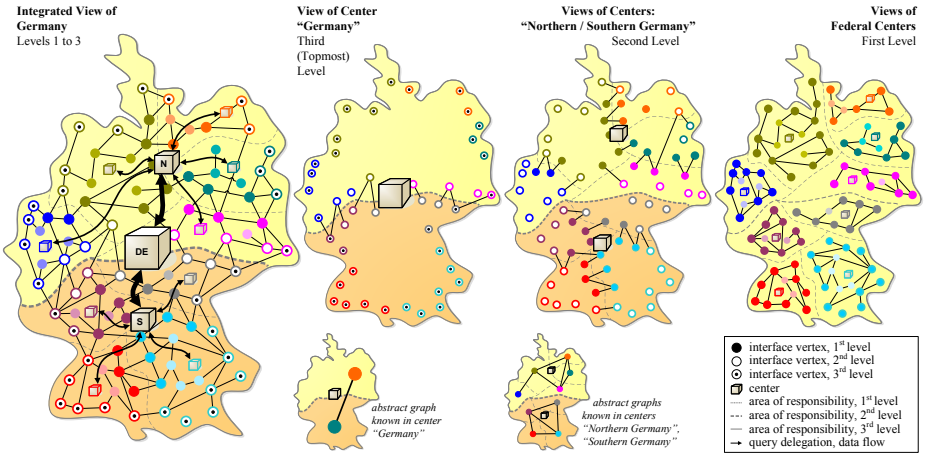

Fig. 1. Exemplary Organization into a Hierarchical Graph

assume that each governed sub-graph G_{c_0} is *path-connected*, i.e., for all nodes $n_1, n_2 \in G_{c_0}$ there exists an internal path in G_{c_0} from n_1 to n_2 (without leaving G_{c_0}).[3] This property will prove useful for the reasoning process later introduced in Section 5.

Note that the structure of centers is, in general, *recursive*: a center c_0 can *either* be responsible for a set S_{c_0} of *subordinate centers* $s_{i \in 0 \ldots n}$, denoted as $c_0 \triangleright s_i$, *or* simply have none. In the former case, c_0 only knows all subordinate centers s_i together with network edges connecting AoRs of these centers, called *joints*. Hence, c_0 *references* some of their nodes but has no knowledge of their internal graph. In contrast, centers without subordinates *manage* nodes (and internal edges). As a consequence, any network element is managed by exactly one center.

To be more precise, each center c_l has a number of *in-* and *out*bound *interface nodes* $I_{c_l}^{c_n}$ and $O_{c_l}^{c_n}$, respectively. They represent points of connection to neighbouring centers c_n and are visible – and, thus, referable – outside of c_l. Interface nodes are depicted in Fig. 1 as round bullets with different filling, depending on the abstraction level.

Moreover, network elements in subordinate centers are (1) jointly exhaustive and (2) pairwise disjoint: (1) $G_{c_0} = (\bigcup_{c_0 \triangleright s_i} G_{s_i}) \cup (\bigcup_{s_i \neq s_j | c_0 \triangleright s_i, c_0 \triangleright s_j} I_{s_i}^{s_j} \times O_{s_i}^{s_j})$; (2) $\forall (s_i \neq s_j | c_0 \triangleright s_i, c_0 \triangleright s_j) : c \triangleright o_i \neq o_j \; G_{s_i} \cap G_{s_j} = \emptyset$. To illustrate this conjunction, consider the left hand side of Fig. 1 depicting the top-level center labelled "DE" (for Germany). There, two smaller centers labelled "N" and "S" (Northern/Southern Germany) are its subordinates. Each of them has, in turn, subordinate centers at the subsequent hierarchy level. The right hand side of Fig. 1 shows three consecutive levels of centers along with joints between their subordinates.

[3] Cases in which G_{c_0} is *not* path-connected can be staved off in a preprocessing step.

The basic elements of a graph G_{c_0} are defined in a core ontology (cf. Fig. 2), which is represented in OWL DL. They are domain-independent and can be extended by more specific classes of a particular application, e.g., for railway networks.

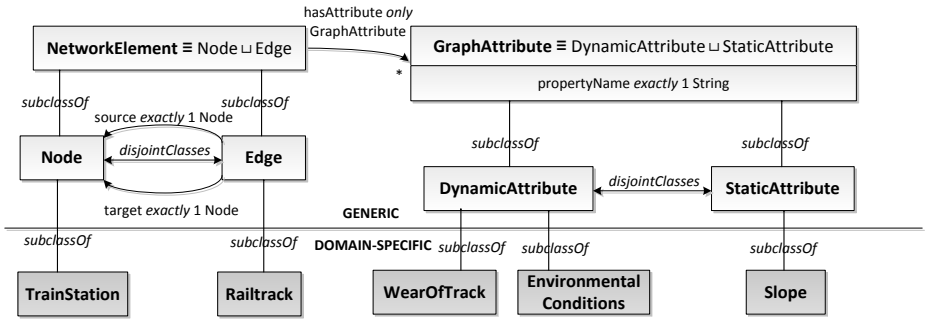

Fig. 2. Core Ontology of a Center

4 Network-based Sensor Data Integration

Central to the integration of sensor measurements is the core ontology (see Fig. 2): Geo-referenced sensor data (e.g., readings of clouds, temperature, etc.) can be assigned to network elements. Assigned sensor data are called *attributes*; they are named. Now the ontology allows for representing all different sorts of attributes, originating from either one or several sensors.[4] The mapping is actually performed by the center whose spatial area (AoR) comprises the current position of a sensor. Otherwise, if a center is adressed which is not responsible for the current sensor measurement (e.g. for train mounted sensors leaving an AoR) the task is delegated to its neighboring centers. Once the correct center is determined, a so-called *impact function* determines the distances between measurements and network elements and subsequently maps them to a weight which expresses the importance of measurements to the individual network element. Using different impact functions, it is possible to model the effect of readings of different sensor types individually, see Fig. 3.

5 Reasoning on Integrated Sensor Data

This section introduces a generic approach to answering queries on the available sensor data, which refer to the topology of a physical infrastructure. We distinguish between different query types and show how complex query types can be

[4] Note that attributes can be aggregated from several primitive sensor measurements involving multiple processing steps.

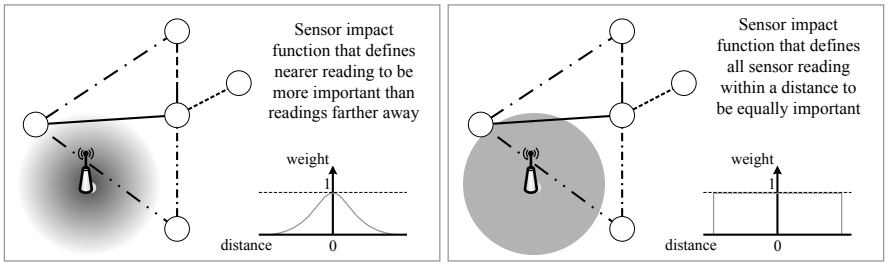

Fig. 3. Assigning Sensor Data to the Network Elements

realized by combinations of a basic query type. This is exemplarily demonstrated for attribute filter and optimal path queries, but can be extended to other query types such as range and k-nearest neighbor queries.

5.1 Basic Query Type: Inspect Queries

An *inspect query* on a network delivers attribute values of a given network element and a given set of attribute names. Attribute names are specified with respect to the core ontology and correspond to specializations of the ontology concept `GraphAttribute` (see Figure 2). For example, `WearOfTrack` could be introduced as a subclass of `DynamicAttribute` and could be further specialized as `WheelImpactLoad`, `TrackGeometry` and `RailProfile`. By querying for `WearOfTrack`, the inspect query would deliver all sensor data, which is available for the given network element and which is semantically relevant for the query according to the ontology.

Answering an inspect query first requires to identify center c containing the given network element i. If the center receiving the query is not identical to c, it delegates the query in a depth-first manner to subordinates (based on the assumption that inspect queries usually refer to network elements in their vicinity). Only if this fails, the query is delegated to its parent and further through the graph hierarchy. Assuming that the network topology is quite static, caching can considerably speed up this process. Once c is identified, the values of the requested attribute types can be retrieved and returned.

5.2 Complex Query Types

Based on appropriate combinations of inspect queries, query answering algorithms for several complex query types such as *attribute filtering*, *optimal path*, and *range queries* (omitted here for brevity) can be realized.

Attribute Filter Queries on a network deliver all network elements that meet a set of given constraints with respect to attribute values and geographical

region. An example is the query for all transformers in a particular region that are currently experiencing overload.

Given a geographical region R, the set of relevant centers can be reduced to those whose AoR intersects R. Each of these centers has to check for each graph element whether the given attibute constraints are met. As up-to-date sensor measurements should be taken into account, preprocessing is usually not possible. However, retrieval of attribute values can be reduced to basic *inspect queries*.

Optimal Path Queries on a network deliver an optimal path connecting two given nodes and optimizing a given optimality criterion. An example is the query for a high-priority freight train route from Hamburg to the customer, where the risk for delay is minimal.

Several algorithms are available for finding or approximating optimal paths in hierarchical graphs [6–9]. The approach proposed in [10] can be extended for finding optimal paths, but requires extensive preprocessing. We therefore propose a refinement-based algorithm, which cannot guarantee an optimal solution, but is able to deal up-to-date measurements and therefore dynamic weights.

Refinement search reduces complexity by abstracting away from the details of a graph: In every center, first an *abstract path* is determined with respect to the *adjacency graph* of its subordinate centers. Then the abstract path is refined to a concrete path.

When searching a near-optimal path between nodes s and t, we assume that centers c_s and c_t (containing s/t) have been identified. Let c_0 be the least common ancestor of c_s and c_t. The *adjacency graph* $G^A_{c_i}$ for a center c_i is defined as $(S_{c_i}, E^A_{c_i})$ with $(c_j, c_k, w) \in E^A_{c_i}, c_j, c_k \in S_{c_i}, w \in \mathbb{N}$ iff c_j, c_k are topologically adjacent (see the bottom of Fig. 1). An *abstract path* $p^{A,s-t}_{c_i}$ from s to t for center c_i is a sequence of centers $(c_j, ..., c_k)$. It is calculated by searching a shortest path from c_j (identical with or governs c_s) to c_k (identical with or governs c_t).

Let $p^{A,s-t}_0 = (c_j, ..., c_k)$ for center c_0. Incremental refinement is performed for all triples (c_{l-1}, c_l, c_{l+1}) in $p^{A,s-t}_{c_0}$:

1. c_0 requests from each $c_l \in p^{A,s-t}_{c_0}$ a weighted shortest path for all pairs $I^{c_l}_{c_{l-1}} \times O^{c_{l+1}}_{c_l}$ of interface nodes. (The special cases are $c_l = c_s$ with $\{s\} \times O^{c_{l+1}}_{c_l}$ and $c_l = c_t$ with $I^{c_l}_{c_{l-1}} \times \{t\}$.) Note that this may require to recursively calculate and refine further abstract paths $p^{A,s-t}_{c_l}$ for each c_l, which can be done in parallel. Weights are query-specific and represent the given optimality criterion. Note that weight calculation can be reduced to *inspect queries* introduced before. This way, path finding takes into account current sensor measurements.
2. Each c_l merges the returned answers with its own graph G_{c_l} into a new temporary graph $G^{s-t}_{c_l}$. It then simply determines a "good" path from s to t by performing a shortest path search on $G^{s-t}_{c_l}$.

It is evident from point two that the quality of the determined path depends on the heuristics used for weighting $p^{A,s-t}_{c_i}$. For example, a simple heuristic, h_1,

assigns to all edges in $G^A_{c_i}$ a uniform weight of 1. Another heuristic, h_2, weights $e \in E^A_{c_i}$ with the minimum of the weights of all edges abstracted by e. Which heuristic leads to a path closer to the optimal path depends on both query and graph. Frequently, however, h_1 is used (see [11]).

Refinement search therefore does not guarantee to find an optimal path. However, it is very efficient in many cases. In particular, it avoids preprocessing and therefore allows to take into account up-to-date sensor measurements.

6 System Architecture and Case Study

We propose the following software design for implementing the previously described reasoning approach and organize the system into several homogeneous software components corresponding to centers. The functionality of a center is decomposed into six sub-components (see Fig. 4):

Sensor Raw Data Store acts as a facade to arbitrary sensor systems and performs network-based sensor data integration as described in Section 4. Thus, it requires access to the graph topology through the interface *IGraphTopology*. It provides access to all sensor readings available for a particular network element via the *ISensorData* interface.

Attribute Provider acts as a facade for arbitrary sensor fusion techniques. It provides convenient access to attribute values for a given network element through the *IObjectAttributes* interface. To this end, it retrieves sensor readings through the *ISensorData* interface and hides the sensor fusion.

Graph Manager provides access to the topology of the subgraph managed by this center through the *IGraphTopology* interface.

Graph Algorithm Provider realizes the graph-based reasoning described in Section 5. It provides the *IPublicGraphSearch* interface to the *Query Processor* and the *IPrivateGraphSearch* to other centers through the *Center Connector*. It requires access to the graph topology through *IGraphTopology* and network element attributes through *IObjectAttributes*.

Query Processor accepts and checks queries as described in Section 5 and delegates them to the *Graph Algorithm Provider*.

Center Connector acts as a proxy to other centers. It provides the *Graph Algorithm Provider* with access to other *Graph Algorithm Provider* through the *IDelegation* interface for center-crossing queries.

We demonstrate the applicability of our approach by a case study in the railway network domain:

The railway network is modelled as a hierarchy of graphs. The root center represents Germany and has two subordinate centers for North Germany and South Germany (see Fig. 1). Both are further partitioned into state and regional networks. In the generic network ontology (see Fig. 2), which is represented in OWL DL, we specialize `Node` to `Station` and `Switch`, among others, which are further specialized to `FreightStation`, `PassengerStation` and so on. `Edge` is

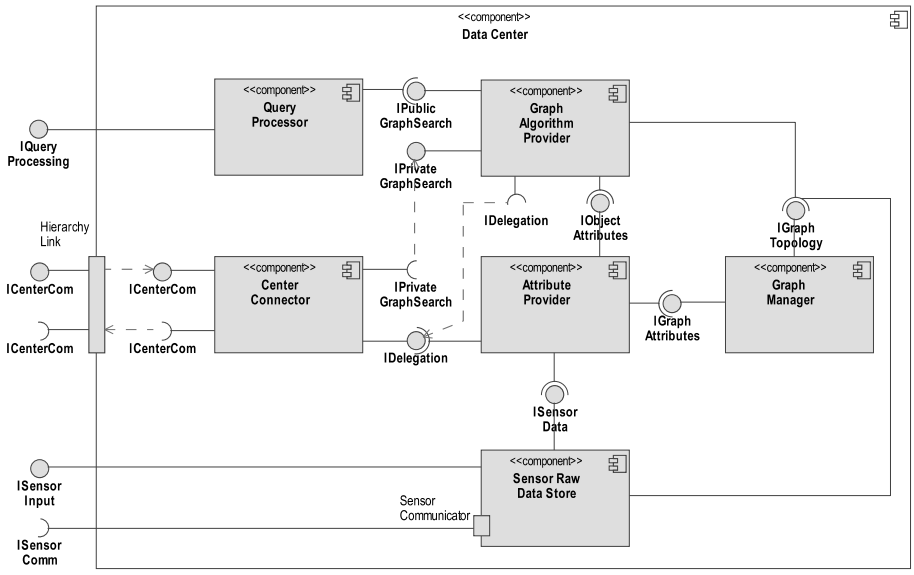

Fig. 4. Software Architecture of a Center

specialized to `Track` with `RailTrack`, `TunnelTrack`, and `BridgeTrack` as subclasses.

Geo-referenced sensor data in this domain are measurements from train-mounted (and therefore mobile) sensor systems as well as data from external sources. Train-mounted systems include track geometry and rail profile sensors, which indicate the wear of a track; external sources include temperature, wind, and precipitation. This information is integrated with respect to the referred network element and represented in RDF format with respect to the OWL DL ontology. Consequently, the overall state of a particular track (e. g. `ReliableTrack`, `PriorityMaintenanceTrack`, or `ImmediateMaintenanceTrack`) can be derived using ontological reasoning.

Finally, queries make use of this information: a traffic manager, who needs to route a booked hazardous goods transport, can identify paths with a maximum level of track quality using an optimal path query. A maintenance manager, who needs to schedule maintenance activities, can retrieve priority maintenance tracks in a certain area using an attribute filter query.

7 Conclusion

This paper presented a domain-independent approach for integrating geo-referenced sensor measurements in physical infrastructures. A spatial network is used as a common spatial reference system; a generic ontology serves as a formal model for sensor semantics, which can be represented using OWL DL. Based on

this, several spatial reasoning tasks with respect to the topology of the physical infrastructure can be supported. In particular, we presented a refinement algorithm for finding near-optimal paths based on basic inspect queries.

We are currently implementing the proposed system architecture and working on hooking up real-world sensor sources from the railway domain. Future work will include performance evaluations as well as the application of the approach to the power network domain.

References

1. General Electrics: Power Management Control System. http://www.geindustrial.com/multilin/enervista/PMCS/ (2007)
2. Salmenjoki, K., Tsaruk, Y., Terziyan, V.Y., Viitala, M.: Agent-based approach for electricity distribution systems. In: Proc. of 9th Int'l Conference on Enterprise Information Systems (ICEIS07). Volume 2., Funchal, Portugal (2007) 382–389
3. Dailey, D., Haselkorn, M., Meyers, D.: A Structured Approach to Developing Real-Time, Distributed NetworkApplications for ITS. In: ITS Journal. (1996)
4. Siemens AG: Siemens Railcom Manager. http://www.transportation.siemens.com (2007)
5. Maly, T., Rumpler, M., Schweinzer, H., Schoebel, A.: New development of an overall train inspection system for increased operational safety. In: Proc. of Intelligent Transportation Systems, IEEE (2005)
6. Holte, R.C., Perez, M.B., Zimmer, R.M., MacDonald, A.J.: Hierarchical a*: Searching abstraction hierarchies efficiently. In: AAAI/IAAI, Vol. 1. (1996) 530–535
7. Holte, R., Drummond, C., Perez, M., Zimmer, R., MacDonald, A.J.: Searching with abstractions: A unifying framework and new high-performance algorithm. In: Canadian Conference on Artificial Intelligence (CAI). (1994) 263–270
8. Yimin, W., Jianmin, X., Yucong, H., Qinghong, Y.: A shortest path algorithm based on hierarchical graph model. In: Intelligent Transportation Systems. (2003)
9. Huang, Y.W., Jing, N., Rundensteiner, E.A.: Hierarchical path views: A model based on fragmentation andtransportation road types. In: ACM-GIS. (1995) 93–
10. Jing, N., Huang, Y.W., Rundensteiner, E.: Hierarchical encoded path views for path query processing:anoptimal model and its performance evaluation. In: IEEE Transactions on Knowledge and Data Engineering. Volume 10. (1998) 409–432
11. Holte, R.C., Mkadmi, T., Zimmer, R.M., MacDonald, A.J.: Speeding up problem solving by abstraction: A graph oriented approach. Artificial Intelligence **85**(1-2) (1996) 321–361