Fusing Approximate Knowledge from Distributed Sources*

Barbara Dunin-Kęplicz and Linh Anh Nguyen and Andrzej Szałas

Abstract In this paper we investigate a technique for fusing approximate knowledge obtained from distributed, heterogeneous information sources. We use a generalization of rough sets and relations [14], which depends on allowing arbitrary similarity relations. The starting point of this research is [2], where a framework for knowledge fusion in multi-agent systems is introduced. Agent's individual perceptual capabilities are represented by similarity relations, further aggregated to express joint capabilities of teams. This aggregation, allowing a shift from individual to social level, has been formalized by means of dynamic logic. The approach of [2] uses the full propositional dynamic logic, not guaranteeing the tractability of reasoning. Therefore the results of [11, 12, 13] are adapted to provide a technical engine for tractable approximate database querying restricted to a Horn fragment of serial PDL. We also show that the obtained formalism is quite powerful in applications.

1 Similarities and Approximate Reasoning

In this paper we investigate a technique for fusing approximate knowledge obtained from distributed information sources. This issue is substantial in modeling multi-agent systems, where a group of loosely coupled heterogeneous agents cooperate in

B. Dunin-Kęplicz
Institute of Informatics, Warsaw University, Poland and ICS, Polish Academy of Sciences, Poland
e-mail: keplicz@mimuw.edu.pl

L.A. Nguyen
Institute of Informatics, Warsaw University, Poland
e-mail: nguyen@mimuw.edu.pl

A. Szałas
Institute of Informatics, Warsaw University, Poland and IDA, Linköping University, Sweden
e-mail: andsz@mimuw.edu.pl

* Supported by the MNiSW grants N N206 399134 and N N206 399334.

achieving a common goal. Information exchange, leading ultimately to knowledge fusion, is a natural and vital ingredient of this process. We use a generalization of rough sets and relations [14], which depends on allowing arbitrary similarity relations, while in [14] only equivalence relations are considered. In order to approximate relations one uses here a covering of the underlying domain by similarity-based neighborhoods. Such approximate relations have been shown to be useful in many application areas requiring the use of approximate knowledge structures [3].

There are many choices as to possible constraints to be placed on the similarity relation used to define approximations. For example, one might not want the relation be transitive since similar objects do not naturally chain in a transitive manner (see, e.g., [1, 6, 3, 10]).

The basic requirement as to approximations is that the lower approximation is included in the upper one of any set/relation. This is equivalent to the seriality of similarity relations (see [5]), which we set as the only requirement.

The focus of this paper is approximate knowledge fusion based on the idea of approximations. Our starting point is [2], where a framework for knowledge fusion in multi-agent systems is introduced. Agent's individual perceptual capabilities are represented by similarity relations, further aggregated to express joint capabilities of teams. The aggregation expressing a shift from individual to social level of agents' activity has been formalized by means of dynamic logic. The approach of [2], as using the full propositional dynamic logic, does not guarantee tractability of reasoning [9]. Our idea is to adapt the techniques of [11, 12, 13] to provide an engine for tractable approximate database querying restricted to a Horn fragment of serial PDL, denoted by SPDL.

To distinguish between extensional and intensional database, we use the terminology of description logics:

- ABox (*assertion box*) stands for the extensional database (containing facts)
- TBox (*terminological box*) stands for the intensional database (containing rules).

In this paper we present an algorithm that, given a Horn query $\mathscr{P}$ expressed in SPDL, and an ABox $\mathscr{A}$, constructs a least SPDL model $\mathscr{M}$ of $\mathscr{P}$ and $\mathscr{A}$. This model has the property that for every positive formula φ and for every individual a, $\varphi(a)$ is a logical consequence of $\mathscr{P}, \mathscr{A}$ in SPDL (denoted by $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$) iff it is true in $\mathscr{M}$ (i.e. $a^{\mathscr{M}} \in \varphi^{\mathscr{M}}$). The least model $\mathscr{M}$ is constructed in polynomial time in the size of $\mathscr{A}$ (and has a polynomial size in the size of $\mathscr{A}$). As a consequence, the problem of checking whether $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$ has PTIME data complexity (measured in the size of $\mathscr{A}$).

The paper is structured as follows. In Section 2 we recall Propositional Dynamic Logic, show its relationship to approximate reasoning and approximate databases, and justify the requirement as to seriality. Section 3 is devoted to showing the PTIME data complexity of the selected Horn fragment of SPDL. Section 4 shows an example illustrating the potential of the introduced machinery in real-world applications.

2 Serial Propositional Dynamic Logic

2.1 Language and Semantics of SPDL

Let us now define *serial propositional dynamic logic* (SPDL). The key idea is to provide calculus on similarity relations rather than on programs. This somehow unusual move allows us to reason about similarities using the whole apparatus of dynamic logic, where "programs" are replaced by similarity relations.

Let $\mathscr{MOD}$ denote the set of *similarity relation symbols*, and $\mathscr{PROP}$ denote the set of *propositions*. We use letters like σ to indicate elements of $\mathscr{MOD}$, and letters like p, q to indicate elements of $\mathscr{PROP}$.

Definition 1. *Formulas* and *similarity expressions* are respectively defined by the two following BNF grammar rules:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi$$
$$\alpha ::= \sigma \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

◁

We use letters like α, β to denote similarity expressions; φ, ψ to denote formulas; and a, b, c to denote *individuals*. Intuitively,

- $\alpha_1;\alpha_2$ stands for a sequential composition of relations α_1 and α_2
- $\alpha_1 \cup \alpha_2$ stands for set-theoretical union of relations α_1 and α_2
- α^* stands for the transitive closure of α
- $\varphi?$ stands for the test operator.

$\langle\alpha\rangle$ and $[\alpha]$ are modal operators of the dynamic logic with the meaning:

- $\langle\alpha\rangle\varphi$: there is an object similar w.r.t. α to a given object and satisfying formula φ
- $[\alpha]\varphi$: all objects similar w.r.t. α to a given object satisfy φ.

The following definitions naturally capture these intuitions. Observe, however, that rather than the set of worlds, the sets of objects are domains of Kripke structures.

Definition 2. A Kripke structure is a pair $\mathscr{M} = \langle \Delta^{\mathscr{M}}, \cdot^{\mathscr{M}} \rangle$, where $\Delta^{\mathscr{M}}$ is a set of *objects*, and $\cdot^{\mathscr{M}}$ is an interpretation function that maps each individual a to an element $a^{\mathscr{M}}$ of $\Delta^{\mathscr{M}}$, each proposition p to a subset $p^{\mathscr{M}}$ of $\Delta^{\mathscr{M}}$, and each similarity relation symbol σ to a binary relation $\sigma^{\mathscr{M}}$ on $\Delta^{\mathscr{M}}$. ◁

The interpretation function is extended for all formulas and similarity expressions:

$$\top^{\mathscr{M}} = \Delta^{\mathscr{M}},\ (\neg\varphi)^{\mathscr{M}} = \Delta^{\mathscr{M}} \setminus \varphi^{\mathscr{M}},\ (\varphi \wedge \psi)^{\mathscr{M}} = \varphi^{\mathscr{M}} \cap \psi^{\mathscr{M}}$$
$$(\varphi \vee \psi)^{\mathscr{M}} = \varphi^{\mathscr{M}} \cup \psi^{\mathscr{M}},\ (\varphi \rightarrow \psi)^{\mathscr{M}} = (\neg\varphi \vee \psi)^{\mathscr{M}}$$
$$(\langle\alpha\rangle\varphi)^{\mathscr{M}} = \{x \in \Delta^{\mathscr{M}} \mid \exists y\, [\alpha^{\mathscr{M}}(x,y) \wedge \varphi^{\mathscr{M}}(y)]\}$$
$$([\alpha]\varphi)^{\mathscr{M}} = \{x \in \Delta^{\mathscr{M}} \mid \forall y\, [\alpha^{\mathscr{M}}(x,y) \rightarrow \varphi^{\mathscr{M}}(y)]\}$$

$$(\alpha;\beta)^{\mathscr{M}} = \alpha^{\mathscr{M}} \circ \beta^{\mathscr{M}} = \{(x,y) \mid \exists z\, [\alpha^{\mathscr{M}}(x,z) \wedge \beta^{\mathscr{M}}(z,y)]\}$$
$$(\alpha \cup \beta)^{\mathscr{M}} = \alpha^{\mathscr{M}} \cup \beta^{\mathscr{M}},\ (\alpha^*)^{\mathscr{M}} = (\alpha^{\mathscr{M}})^*,\ (\varphi?)^{\mathscr{M}} = \{(x,x) \mid \varphi^{\mathscr{M}}(x)\}.$$

We sometimes write $\mathscr{M},x \models \varphi$ to denote $x \in \varphi^{\mathscr{M}}$. For a set Γ of formulas, we write $\mathscr{M},x \models \Gamma$ to denote that $\mathscr{M},x \models \varphi$ for all $\varphi \in \Gamma$. If $\mathscr{M},x \models \Gamma$ for all $x \in \Delta^{\mathscr{M}}$ then we call $\mathscr{M}$ a *model of* Γ.

When dealing with the data complexity of the instance checking problem, without loss of generality we can assume that both the sets $\mathscr{MOD}$ and $\mathscr{PROP}$ are finite and fixed. Under this assumption, the *size of a Kripke structure* $\mathscr{M}$ is defined to be the number of elements of the set $\Delta^{\mathscr{M}}$.

For every $\sigma \in \mathscr{MOD}$, we adopt the *seriality axioms* $[\sigma]\varphi \rightarrow \langle\sigma\rangle\varphi$. It is well known (see, e.g., [5]) that seriality axioms correspond to the *seriality property* $\forall x \exists y [\sigma^{\mathscr{M}}(x,y)]$. By an *admissible interpretation for* SPDL we understand any Kripke structure $\mathscr{M}$ with all $\sigma \in \mathscr{MOD}$ satisfying the seriality property. We call such Kripke structures *serial*.

2.2 SPDL as a Query Language in Approximate Databases

Let us now explain how SPDL is used as a query language. First observe that interpretations assign sets of objects to formulas. Thus it is natural to consider each formula as the query selecting all objects satisfying the formula.

Example 1. Let, in a given interpretation $\mathscr{M}$, $\Delta^{\mathscr{M}} = \{o_1,o_2,o_3,o_4,o_5\}$, $red^{\mathscr{M}} = \{o_1,o_3,o_4\}$ and $small^{\mathscr{M}} = \{o_1,o_2,o_4,o_5\}$. Then the $(red \wedge small)^{\mathscr{M}} = \{o_1,o_4\}$, thus the query $(red \wedge small)$ returns the set $\{o_1,o_4\}$. Similarly, the query $(red \rightarrow small)$ returns $\{o_1,o_2,o_4,o_5\}$. ◁

Let us now recall the notion of approximations.

Definition 3. Let Δ be a set of objects and α be a similarity expression representing a serial binary relation on Δ. For $a \in \Delta$, by the *neighborhood of a w.r.t.* α, we understand the set of elements similar to a, $n^{\alpha}(a) \stackrel{\text{def}}{=} \{b \in \Delta \mid \alpha(a,b)\}$. For $A \subseteq \Delta$, the *lower and upper approximations of A w.r.t.* α, denoted respectively by A^{+}_{α} and $A^{\oplus}_{\alpha}$, are defined by $A^{+}_{\alpha} = \{a \in \Delta \mid n^{\alpha}(a) \subseteq A\}$, $A^{\oplus}_{\alpha} = \{a \in \Delta \mid n^{\alpha}(a) \cap A \neq \emptyset\}$. ◁

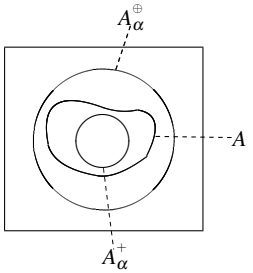

Fig. 1 Lower approximation A^{+}_{α} and upper approximation $A^{\oplus}_{\alpha}$ of a set A.

The meaning of those approximations is illustrated in Figure 1. Intuitively, assuming that the perception of an agent is modeled by similarity expression α,

- $a \in A^{+}_{\alpha}$ means that, from the point of view of the agent, a surely is in A, since all objects indiscernible from a are in A
- $a \in A^{\oplus}_{\alpha}$ means that, from the point of view of the agent, a possibly is in A, since there are objects indiscernible from a which are in A.

Now observe that:

$$[\alpha]A \text{ expresses the lower approximation of } A \text{ w.r.t. } \alpha, \text{ i.e., } A^{+}_{\alpha}, \tag{1}$$

$$\langle\alpha\rangle A \text{ expresses the upper approximation of } A \text{ w.r.t. } \alpha, \text{ i.e., } A^{\oplus}_{\alpha}. \tag{2}$$

Remark 1. In the view of (1) and (2), the seriality axiom expresses the property that the lower approximation of a set w.r.t. any similarity expression is included in its upper approximation. This justifies seriality as the basic requirement on approximations. ◁

Example 2. Let $\mathscr{M}$ be the interpretation considered in Example 1. Let σ be the reflexive closure of relation $\{\langle o_1,o_2\rangle,\langle o_2,o_1\rangle,\langle o_3,o_4\rangle\}$. Then, for example,

$$red^{+}_{\sigma} = \{o_3,o_4\},\ red^{\oplus}_{\sigma} = \{o_1,o_2,o_3,o_4\}.$$ ◁

2.3 The Horn Fragment of SPDL

In order to express tractable queries we restrict the query language to the Horn fragment.

Definition 4. *Positive formulas* φ_{pos} are defined by the following BNF grammar:

$$\begin{aligned}
\varphi_{pos} &::= \top \mid p \mid \varphi_{pos} \wedge \varphi_{pos} \mid \varphi_{pos} \vee \varphi_{pos} \mid \langle\alpha_{pos_\Diamond}\rangle\varphi_{pos} \mid [\alpha_{pos_\Box}]\varphi_{pos} \\
\alpha_{pos_\Diamond} &::= \sigma \mid \alpha_{pos_\Diamond};\alpha_{pos_\Diamond} \mid \alpha_{pos_\Diamond} \cup \alpha_{pos_\Diamond} \mid \alpha^{*}_{pos_\Diamond} \mid \varphi_{pos}? \\
\alpha_{pos_\Box} &::= \sigma \mid \alpha_{pos_\Box};\alpha_{pos_\Box} \mid \alpha_{pos_\Box} \cup \alpha_{pos_\Box} \mid \alpha^{*}_{pos_\Box} \mid (\neg\varphi_{pos})?
\end{aligned}$$

Program clauses φ_{prog} are defined by the following BNF grammar:[2]

$$\begin{aligned}
\varphi_{prog} &::= \top \mid p \mid \varphi_{pos} \rightarrow \varphi_{prog} \mid \varphi_{prog} \wedge \varphi_{prog} \mid \langle\alpha_{prog_\Diamond}\rangle\varphi_{prog} \mid [\alpha_{prog_\Box}]\varphi_{prog} \\
\alpha_{prog_\Diamond} &::= \sigma \mid \alpha_{prog_\Diamond};\alpha_{prog_\Diamond} \mid \varphi_{prog}? \\
\alpha_{prog_\Box} &::= \sigma \mid \alpha_{prog_\Box};\alpha_{prog_\Box} \mid \alpha_{prog_\Box} \cup \alpha_{prog_\Box} \mid \alpha^{*}_{prog_\Box} \mid \varphi_{pos}?
\end{aligned}$$

A *positive logic program* is a finite set of program clauses. ◁

Example 3. Observe that the Horn fragment of SPDL is quite expressive. For example, it allows one to express a variant of default rules (discussed, e.g., in [3]). Namely, a typical default rule can be expressed as $A^{+}_{\sigma},B^{\oplus}_{\sigma} \vdash C^{+}_{\sigma}$, with intuitive meaning "if A is surely true and B might be true then accept C as surely true". ◁

Let us now formally link SPDL with databases.

[2] Notice the two occurrences of φ_{pos} in the grammar. We do not allow formulas of the form $\langle\alpha\cup\beta\rangle\varphi$ or $\langle\alpha^{*}\rangle\varphi$ to be program clauses because they cause non-determinism.

Definition 5. An *individual assertion* is an expression of the form $p(a)$, where p is a proposition and a is an individual. A *similarity assertion* is an expression of the form $\sigma(a,b)$, where σ is a similarity relation symbol and a, b are individuals. An *ABox* is a finite set of individual assertions and similarity assertions. ◁

Comparing to description logics, individual assertions play a role of concept assertions, while similarity assertions play a role of assertions.

Definition 6.

- Given a Kripke structure $\mathscr{M}$ and an ABox $\mathscr{A}$, we say that $\mathscr{M}$ is a *model of* $\mathscr{A}$, denoted by $\mathscr{M} \models \mathscr{A}$, if $a^{\mathscr{M}} \in p^{\mathscr{M}}$ for every individual assertion $p(a) \in \mathscr{A}$ and $(a^{\mathscr{M}}, b^{\mathscr{M}}) \in \sigma^{\mathscr{M}}$ for every similarity assertion $\sigma(a,b) \in \mathscr{A}$.
- Given a positive logic program $\mathscr{P}$, an ABox $\mathscr{A}$, a positive formula φ and an individual a, we say that *a has the property φ w.r.t. $\mathscr{P}$ and $\mathscr{A}$ in* SPDL (or $\varphi(a)$ is a logical consequence of $\mathscr{P}, \mathscr{A}$ in SPDL), denoted by $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$, if for every serial Kripke structure $\mathscr{M}$, if $\mathscr{M}$ is a model of $\mathscr{P}$ and $\mathscr{A}$ then $a^{\mathscr{M}} \in \varphi^{\mathscr{M}}$.
- By the *instance checking* problem of the Horn fragment of SPDL we mean the problem of checking $\mathscr{P}, \mathscr{A} \models_s \varphi(a)$. The *data complexity* of this problem is measured when $\mathscr{P}$, φ and a are fixed (and compose a query), while $\mathscr{A}$ varies as input data. ◁

3 Computational Aspects

A Kripke structure $\mathscr{M}$ is *less than or equal to* $\mathscr{M}'$, $\mathscr{M} \leq \mathscr{M}'$, if for every positive formula φ and every individual a, $a^{\mathscr{M}} \in \varphi^{\mathscr{M}}$ implies $a^{\mathscr{M}'} \in \varphi^{\mathscr{M}'}$.

Let $\mathscr{P}$ be a positive logic program and $\mathscr{A}$ be an ABox. We say that a Kripke structure $\mathscr{M}$ is a *least* SPDL *model of $\mathscr{P}$ and $\mathscr{A}$* if $\mathscr{M}$ is an SPDL model of $\mathscr{P}$ and $\mathscr{A}$ and is less than or equal to any other SPDL model of $\mathscr{P}$ and $\mathscr{A}$.

Let us now present an algorithm that, given a positive logic program $\mathscr{P}$ and an ABox $\mathscr{A}$, constructs a finite least SPDL model of $\mathscr{P}$ and $\mathscr{A}$. The algorithm constructs the following data structures:

- Δ is a set of objects. We distinguish the subset Δ_0 of Δ that consists of all the individuals occurring in the ABox $\mathscr{A}$. In the case $\mathscr{A}$ is empty, let $\Delta_0 = \{\tau\}$ for some element τ.
- H is a mapping that maps every $x \in \Delta$ to a set of formulas, which are the properties that should hold for x. When the elements of Δ are treated as states, $H(x)$ denotes the content of the state x.
- *Next* is a mapping such that, for $x \in \Delta$ and $\langle\sigma\rangle\varphi \in H(x)$, we have $Next(x, \langle\sigma\rangle\varphi) \in \Delta$. The meaning of $Next(x, \langle\sigma\rangle\varphi) = y$ is that:
  - $\langle\sigma\rangle\varphi \in H(x)$ and $\varphi \in H(y)$,
  - the "requirement" $\langle\sigma\rangle\varphi$ is realized for x by going to y via a σ-transition.

Using the above data structures, we define a Kripke structure $\mathscr{M}$ such that:

- $\Delta^{\mathscr{M}} = \Delta$,
- $a^{\mathscr{M}} = a$ for every individual a occurring in $\mathscr{A}$,
- $p^{\mathscr{M}} = \{x \in \Delta \mid p \in H(x)\}$ for every $p \in \mathscr{PRCP}$,
- $\sigma^{\mathscr{M}} = \{(a,b) \mid \sigma(a,b) \in \mathscr{A}\} \cup \{(x,y) \mid Next(x, \langle\sigma\rangle\varphi) = y \text{ for some } \varphi\}$ for every $\sigma \in \mathscr{MCD}$.

The *saturation* of a set Γ of formulas, denoted by $\mathsf{Sat}(\Gamma)$, is defined to be the smallest superset of Γ such that:

- $\top \in \mathsf{Sat}(\Gamma)$ and $\langle\sigma\rangle\top \in \mathsf{Sat}(\Gamma)$ for all $\sigma \in \mathscr{MCD}$,
- if $\varphi \wedge \psi \in \mathsf{Sat}(\Gamma)$ or $\langle\varphi?\rangle\psi \in \mathsf{Sat}(\Gamma)$ then $\varphi \in \mathsf{Sat}(\Gamma)$ and $\psi \in \mathsf{Sat}(\Gamma)$,
- if $\langle\alpha;\beta\rangle\varphi \in \mathsf{Sat}(\Gamma)$ then $\langle\alpha\rangle\langle\beta\rangle\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha;\beta]\varphi \in \mathsf{Sat}(\Gamma)$ then $[\alpha][\beta]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha \cup \beta]\varphi \in \mathsf{Sat}(\Gamma)$ then $[\alpha]\varphi \in \mathsf{Sat}(\Gamma)$ and $[\beta]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha^*]\varphi \in \mathsf{Sat}(\Gamma)$ then $\varphi \in \mathsf{Sat}(\Gamma)$ and $[\alpha][\alpha^*]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\varphi?]\psi \in \mathsf{Sat}(\Gamma)$ then $(\varphi \to \psi) \in \mathsf{Sat}(\Gamma)$.

The *transfer of Γ through σ* is defined by $\mathsf{Trans}(\Gamma,\sigma) \stackrel{\text{def}}{=} \mathsf{Sat}(\{\varphi \mid [\sigma]\varphi \in \Gamma\})$. We use procedure $\mathsf{Find}(\Gamma)$ defined as:

> if there exists $x \in \Delta \setminus \Delta_0$ with $H(x) = \Gamma$ then return x,
> else add a new object x to Δ with $H(x) = \Gamma$ and return x.

The algorithm shown in Figure 2 constructs a least SPDL model for a positive logic program $\mathscr{P}$ and an ABox $\mathscr{A}$.

Theorem 1.

1. *Let $\mathscr{P}$ be a positive logic program and $\mathscr{A}$ be an ABox. The Kripke structure $\mathscr{M}$ constructed by the algorithm shown in Figure 2 for $\mathscr{P}$ and $\mathscr{A}$ is a least* SPDL *model of $\mathscr{P}$ and $\mathscr{A}$.*
2. *Let $\mathscr{P}$ be a positive logic program, $\mathscr{A}$ an ABox, φ a positive formula, and a an individual. Then checking $(\mathscr{P},\mathscr{A}) \models_s \varphi(a)$ can be done in polynomial time in the size of $\mathscr{A}$ (by constructing a least* SPDL *model $\mathscr{M}$ of $\mathscr{P}$ and $\mathscr{A}$ using our algorithm, and then checking whether $a^{\mathscr{M}} \in \varphi^{\mathscr{M}}$). That is, the data complexity of the Horn fragment of* SPDL *is in* PTIME. ◁

4 An Exemplary Scenario

Consider the following scenario:

> Two robots, R_1 and R_2, have the goal to move objects from one place to another. Each robot is able to move objects of a specific signature,[3] and together they might be able to move objects of a combined signature. Some objects, when attempted to be moved, may cause some damages for robots. Robots are working independently, but sometimes have to cooperate to achieve their goals.

To design such robots one has to make a number of decisions as described below.

[3] For example, dependent on weight, size and type of surface.

Input: A positive logic program $\mathscr{P}$ and an ABox $\mathscr{A}$.
Output: A least SPDL model $\mathscr{M}$ of $\mathscr{P}$ and $\mathscr{A}$.

1. let Δ_0 be the set of all individuals occurring in $\mathscr{A}$;
   if $\Delta_0 = \emptyset$ then $\Delta_0 := \{\tau\}$;
   set $\Delta := \Delta_0$, $\mathscr{P}' := \mathsf{Sat}(\mathscr{P})$;
   for $x \in \Delta$, set $H(x) := \mathscr{P}' \cup \{p \mid p(x) \in \mathscr{A}\}$;
2. for every $x \in \Delta$ reachable from Δ_0 and for every formula $\varphi \in H(x)$
   a. case $\varphi = \langle\sigma\rangle\psi$: if $Next(x, \langle\sigma\rangle\psi)$ is not defined then
      $Next(x, \langle\sigma\rangle\psi) := \mathsf{Find}(\mathsf{Sat}(\{\psi\}) \cup \mathsf{Trans}(H(x), \sigma) \cup \mathscr{P}')$;
   b. case $\varphi = [\sigma]\psi$:
      i. for every $y \in \Delta_0$ such that $\sigma^{\mathscr{M}}(x, y)$ holds and $\psi \notin H(y)$
         $H(y) := H(y) \cup \mathsf{Sat}(\{\psi\})$;
      ii. for every $y \in \Delta \setminus \Delta_0$ such that $\sigma^{\mathscr{M}}(x, y)$ holds and $\psi \notin H(y)$
         A. $y_* := \mathsf{Find}(H(y) \cup \mathsf{Sat}(\{\psi\}))$;
         B. for every ξ such that $Next(x, \langle\sigma\rangle\xi) = y$
            $Next(x, \langle\sigma\rangle\xi) := y_*$;
   c. case $\varphi = (\psi \to \xi)$: if $x \in \psi^{\mathscr{M}}$ and $Next(y, \langle\sigma\rangle\top)$ is defined for every y reachable from x and every $\sigma \in \mathscr{MCD}$ then
      i. if $x \in \Delta_0$ then $H(x) := H(x) \cup \mathsf{Sat}(\{\xi\})$
      ii. else
         A. $x_* := \mathsf{Find}(H(x) \cup \mathsf{Sat}(\{\xi\}))$;
         B. for every y, σ, ζ such that $Next(y, \langle\sigma\rangle\zeta) = x$
            $Next(y, \langle\sigma\rangle\zeta) := x_*$;
3. if some change occurred, go to Step 2;
4. delete from Δ every x unreachable from Δ_0 and delete from H and $Next$ all elements related with such an x.

Fig. 2 Algorithm constructing the least SPDL model for a positive logic program and an ABox.

4.1 Formalizing Movability of Objects

We assume that the signature of movable objects for each robot is given by its specification together with a similarity relation defining the range of these objects as follows:

$$spec_1 \stackrel{\text{def}}{\equiv} (light \wedge smooth) \vee (heavy \wedge rough) - \text{ for robot } R_1 \tag{3}$$

$$spec_2 \stackrel{\text{def}}{\equiv} small \vee medium - \text{ for robot } R_2. \tag{4}$$

Movable objects are then specified by

$$spec_i \to movable_i, \tag{5}$$

where $i \in \{1,2\}$ and $movable_i$ is true for objects that can be moved by R_i.

The idea is that all objects similar to movable ones are movable too.[4] Let σ_1 and σ_2 be similarity relations reflecting perceptual capabilities of R_1 and R_2, respectively

[4] This is a very natural and quite powerful technique, allowing one to express the inheritance of particular properties of objects by similar objects.

(for a discussion of such similarity relations based on various sensor models see [4]). Now, in addition to (5), movable objects are characterized by

$$\langle \sigma_i \rangle spec_i \rightarrow movable_i. \tag{6}$$

Remark 2. Note that rather than (6) one could assume $[\sigma_i]spec_i \rightarrow movable_i$. This would mean that, in addition to (5), we would consider objects which are similar only to movable objects. In some applications this choice would indeed be reasonable and perhaps less risky. ◁

Observe that in general it is impossible to automatically derive combined signatures that specify what robots can move together. Therefore, we introduce $spec_3$ and α_3 as a specification of such joint capabilities. An example of $spec_3$ can be given by

$$spec_3 \stackrel{\text{def}}{\equiv} large \wedge rough. \tag{7}$$

Objects movable by robots working together are then defined by

$$(spec_3 \vee \langle \alpha_3 \rangle spec_3) \rightarrow movable\_by\_two. \tag{8}$$

Observe that α_3 is usually computed on the basis of σ_1 and σ_2, since we do not assume any observer other than R_1, R_2, and σ_1, σ_2 reflects their perceptual capabilities. We shall assume that

$$\alpha_3 \stackrel{\text{def}}{=} \sigma_1 \cup \sigma_2 \cup (\sigma_1; \sigma_2) \cup (\sigma_2; \sigma_1). \tag{9}$$

The meaning of this expression is that an object o is similar w.r.t. α_3 to o' whenever

- o is perceived similar to o' by R_1 or by R_2, or
- there is an object o'' such that
  - o is perceived similar to o'' by R_1 and o'' is perceived similar to o' by R_2, or
  - o is perceived similar to o'' by R_2 and o'' is perceived similar to o' by R_1.

Let us emphasize that one can use much more complex expressions, reflecting particular algorithms for computing α_3, since our operators are those accepted in PDL. Of course, there are some restrictions, if one wants to stay in a tractable framework. Recall that we have accepted Definition 4 to guarantee tractability.

4.2 The Database

Let $\mathscr{A}$ be the ABox consisting of the assertions about objects $o_1, \ldots, o_5$ that are illustrated in Figure 3. It contains, for example, $light(o_1)$, $smooth(o_1)$, $\sigma_1(o_1, o_2)$, $\sigma_1(o_2, o_1)$, etc. Let $\mathscr{P}$ be the positive logic program consisting of the clauses (5) and (6) for $i \in \{1, 2\}$, and (8). We consider here the database consisting of $\mathscr{P}$ and $\mathscr{A}$ in SPDL (which adopts the axioms $\langle \sigma_1 \rangle \top$ and $\langle \sigma_2 \rangle \top$ for all the objects of the domain).

In Figure 4 we present the least SPDL model $\mathscr{M}$ of $\mathscr{P}$ and $\mathscr{A}$ constructed by the algorithm. The object o_6 is the only additional object, not satisfying any proposition.

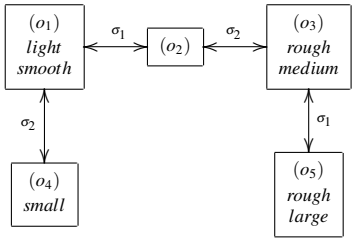

Fig. 3 An ABox for the example considered in Section 4.2.

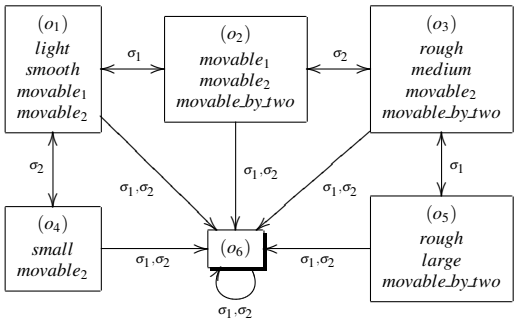

Fig. 4 The least SPDL model of the database considered in Section 4.2.

4.3 Some Queries

Recall that, having a least SPDL model $\mathscr{M}$ of $\mathscr{P}$ and $\mathscr{A}$, to check whether an individual a has a positive property φ w.r.t. $\mathscr{P}$ and $\mathscr{A}$ in SPDL, it suffices to check whether $a \in \varphi^{\mathscr{M}}$. We have that $movable_1^{\mathscr{M}} = \{o_1, o_2\}$, $movable_2^{\mathscr{M}} = \{o_1, o_2, o_3, o_4\}$, $movable\_by\_two^{\mathscr{M}} = \{o_2, o_3, o_5\}$, $(movable\_by\_two \wedge \langle\sigma_1\rangle movable_1)^{\mathscr{M}} = \{o_1\}$, $(movable\_by\_two \wedge [\sigma_1] movable_1)^{\mathscr{M}} = \emptyset$.

5 Conclusions

In this paper we have presented a powerful formalism for approximate knowledge fusion, based on adaptation of Propositional Dynamic Logic. We have shown that restricting this logic to its suitably chosen Horn fragment results in tractable querying mechanism which can be applied in application, where approximate knowledge from various sources is to be fused, e.g., in robotics and multi-agent systems.

Importantly, serial PDL, denoted by SPDL, is also useful as a description logic for domains where seriality condition appears naturally.[5] For example, in reasoning about properties of web pages one can assume that every considered web page has a link to another page (or to itself).

We plan to extend the framework to deal with other operations on similarity relations, which would allow expressing even more subtle approximations and fused knowledge structures applicable in different stages of teamwork in multi-agent systems as discussed in [7, 8].

References

1. S.P. Demri and E.S. Orłowska. *Incomplete Information: Structure, Inference, Complexity*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Heidelberg, 2002.
2. P. Doherty, B. Dunin-Kęplicz, and A. Szałas. Dynamics of approximate information fusion. In M. Kryszkiewicz, J. Peters, H. Rybinski, and A. Skowron, editors, *Proceedings of RSEISP'2007, LNAI 4585*, pages 668–677. Springer-Verlag, 2007.
3. P. Doherty, W. Łukaszewicz, A. Skowron, and A. Szałas. *Knowledge Representation Techniques. A Rough Set Approach*, volume 202 of *Studies in Fuziness and Soft Computing*. Springer-Verlag, 2006.
4. P. Doherty, W. Łukaszewicz, and A. Szałas. Communication between agents with heterogeneous perceptual capabilities. *Journal of Information Fusion*, 8(1):56–69, 2007.
5. P. Doherty and A. Szałas. On the correspondence between approximations and similarity. In S. Tsumoto, R. Slowinski, J. Komorowski, and J.W. Grzymala-Busse, editors, *Proceedings of RSCTC'2004, LNAI 3066*, pages 143–152. Springer-Verlag, 2004.
6. B. Dunin-Kęplicz and A. Szałas. Towards approximate BGI systems. In H-D. Burkhard, G. Lindeman, L. Varga, and R. Verbrugge, editors, *Proceedings of CEEMAS'2007, LNAI 4696*, pages 277–287. Springer-Verlag, 2007.
7. B. Dunin-Kęplicz and R. Verbrugge. Collective intentions. *Fundamenta Informaticae*, 51(3):271–295, 2002.
8. B. Dunin-Kęplicz and R. Verbrugge. A tuning machine for cooperative problem solving. *Fundamenta Informaticae*, 63:283–307, 2004.
9. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
10. J. Maluszyński, A. Szałas, and A. Vitória. Paraconsistent logic programs with four-valued rough sets. In C.-C. Chan, J. Grzymala-Busse, and W. Ziarko, editors, *Proceedings of RSCTC'2008, LNAI 5306*, pages 41–51, 2008.
11. L.A. Nguyen. On the deterministic Horn fragment of test-free PDL. In I. Hodkinson and Y. Venema, editors, *Advances in Modal Logic - Volume 6*, pages 373–392. King's College Publications, 2006.
12. L.A. Nguyen. Weakening Horn knowledge bases in regular description logics to have PTIME data complexity. In S. Ghilardi, U. Sattler, V. Sofronie-Stokkermans, and A. Tiwari, editors, *Proceedings of ADDCT'2007*, pages 32–47, 2007.
13. L.A. Nguyen. Constructing finite least Kripke models for positive logic programs in serial regular grammar logics. *Logic Journal of the IGPL*, 16(2):175–193, 2008.
14. Z. Pawlak. *Rough Sets. Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, 1991.

[5] Related work on Horn fragments of description logics can be found in [12]. The papers by other authors do not consider PDL but only versions of the description logic $\mathscr{SHI}$.