A Logical Account of Causal and Topological Maps

Emilio Remolina and Benjamin Kuipers[1]
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

Technical Report: AI01-288
April 2001

[1]This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by NSF grants IRI-9504138 and CDA 9617327, and by funding from Tivoli Corporation.

A Logical Account of Causal and Topological Maps

Emilio Remolina and Benjamin Kuipers[2]
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

Technical Report: AI01-288
April 2001

Abstract

We consider the problem of how an agent creates a discrete spatial representation from its continuous interactions with the environment. Such representation will be the *minimal* one that explains the experiences of the agent in the environment. In this paper we take the Spatial Semantic Hierarchy as the agent's target spatial representation, and use a circumscriptive theory to specify the minimal models associated with this representation. We provide a logic program to calculate the models of the proposed theory. We also illustrate how the different levels of the representation assume different spatial properties about both the environment and the actions performed by the agent. These spatial properties play the role of "filters" the agent applies in order to distinguish the different environment states it has visited.[3]

1 Introduction

The problem of map building –how an agent creates a discrete spatial representation from its continuous interactions with the environment– can be stated formally as an abduction task where the actions and observations of the agent are explained by connectivity relations among places in the environment [Shanahan, 1996, Shanahan, 1997, Remolina and Kuipers, 1998]. In this paper we consider the Spatial Semantic Hierarchy (SSH) [Kuipers, 2000, Kuipers and Byun, 1988, Kuipers and Byun, 1991] as the agent's target spatial representation. The SSH is a set of distinct representations for large scale space, each with its own ontology and each abstracted from the levels below it. The SSH describes the different states of knowledge that an agent uses in order to organize its sensorimotor experiences and create a spatial representation (i.e. a map). Using the SSH representation, navigation among places is not dependent on the accuracy, or even the existence, of metrical knowledge of the environment.

In order to define the *preferred models* associated with the experiences of the agent, we use a circumscriptive theory to specify the SSH's (minimal) models. Different models can exist that explain the same set of experiences. This occurs because the agent

[2]This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by NSF grants IRI-9504138 and CDA 9617327, and by funding from Tivoli Corporation.

[3]This report is the extended version of the paper [Remolina and Kuipers, 2001] to appear in IJCAI-01, August, 20001.

could associate the same sensory description to different environment states, or because the agent has not completely explored the environment. The different SSH levels assume different spatial properties about the environment and the actions performed by the agent. These spatial properties play the role of "filters" the agent applies in order to distinguish the different environment states it has visited. For instance, at the SSH causal level two environment states are considered the same if any sequence of actions started at these states renders the same sequence of observations. At the SSH topological level, two environment states are considered the same if they are at the same place along the same paths. Finally, at the SSH metrical level, two environment states are the same, if it is possible to assign to them the same coordinates in any frame of reference available to the agent. In sections 3 and 4 we make precise the claims above.

Finally, we use the SSH circumscriptive theory as the specification for a logic program used to implement the abduction task. In the paper we provide the logic program for the SSH causal level theory and illustrate how to encode the minimality condition associated with this theory. We have implemented the program using Smodels [Niemelä and Simons, 1997] and confirm that the theory yields the intended models.

2 Related Work

The SSH grew out of the TOUR model proposed in [Kuipers, 1977, Kuipers, 1978]. Other computational theories of the cognitive map have been proposed: [Kortenkamp *et al.*, 1995, McDermott and Davis, 1984, Leiser and Zilbershatz, 1989, Yeap, 1988]. These theories share the same basic principles: the use of multiple frames of reference, qualitative representation of metrical information, and connectivity relations among landmarks. They differ in how they define what a landmark is, or the description (view, local 2D geometry) associated with a landmark. Except for McDermott and Davis, none of the theories above has a formal account like the one presented in this paper for the SSH.

Considering map building as a formal abduction task has been proposed by Shanahan [1996, 1997]. He proposes a logic-based framework (based on the circumscriptive event calculus) in which a robot constructs a model of the world through an abductive process whereby sensor data is explained by hypothesizing the existence, locations, and shapes of objects. In Shanahan's work, space is considered a real-valued coordinate system. As pointed out in [Shanahan, 1997], a problem of Shanahan's approach is the existence of many minimal models (maps) that explain the agent's experiences. We have alleviated this problem by considering the SSH topological map instead of an Euclidean space as the agent's target spatial representation.

The problem of distinguishing environment states by outputs (views) and inputs (actions) has been studied in the framework of automata theory [Basye *et al.*, 1995]. In this framework, the problem we address here is the one of finding the smallest automaton (w.r.t. the number of states) consistent with a given set of input/output pairs. Without any particular assumptions about the environment or the agent's perceptual abilities, the problem of finding this smallest automaton is NP-complete [Basye *et al.*, 1995].

The SSH [Kuipers, 2000, Kuipers and Byun, 1988, Kuipers and Byun, 1991] abstracts the structure of an agent's spatial knowledge in a way that is relatively independent of its sensorimotor apparatus and the environment within which it moves. At the *SSH control level*, the agent and its environment are modeled as continuous dynamical systems whose equilibrium points are abstracted to a discrete set of *distinctive states*. A distinctive state has associated a *view* describing the sensory input obtained at that distinctive state. The control laws, whose executions define trajectories linking these distinctive states, are abstracted to *actions*, giving a discrete causal graph representation for the state space. The causal graph of states and actions can in turn be abstracted to a topological network of *places*, *paths* and *regions* (i.e. the *topological map*). Local metrical models, such as occupancy grids, of neighborhoods of places and paths can then be built on the framework of the topological network while avoiding global metrical consistency problems. In the next sections we formally describe the SSH causal and topological levels.

3 SSH Causal level

We use a first order sorted language in order to describe the SSH causal level. The sorts of this language include *distinctive states*, *views*, *actions* and *schemas*. The sort of distinctive states corresponds to the names given by the agent to the fixpoints of hill-climbing control strategies. It is possible for the agent to associate different distinctive state names with the same environment state. This is the case since the agent might not know at which of several environment states it is currently located. A distinctive state has an associated view. We use the predicate $View(ds, v)$ to represent the fact that *v* is a *view* associated with *distinctive state* ds. We assume that a distinctive state has a unique view. However, we do **not** assume that views uniquely determine distinctive states (i.e. $View(ds, v) \wedge View(ds', v) \not\rightarrow ds = ds'$). This is the case since the sensory capabilities of an agent may not be sufficient to distinguish distinctive states.

An action has a unique type, either *travel* or *turn*, associated with it. We use the predicate $Action\_type(a, type)$ to represent the fact that the type of action *a* is *type*. Turn actions have associated a unique turn description, either *turnLeft*, *turnRight* or *turnAround*. We use the predicate $Turn\_desc(a, desc)$ to indicate that *desc* is the turn description associated with the turn action *a*.

A schema represents an action execution performed by the agent in the environment. An action execution is characterized in terms of the distinctive states the agent was at before and after the action was performed.[4] We use the predicate $CS(s, ds, a, ds')$ to denote the fact that according to schema s, action a was performed starting at distinctive state ds and ending at distinctive state ds'. While schemas are explicit objects of our theory, most of the time it is convenient to leave them implicit. We introduce the following convenient notation:

$$\langle ds, a, ds' \rangle \equiv_{def} \exists s \; CS(s, ds, a, ds')$$

$$\langle ds, type, ds' \rangle \equiv_{def} \exists a \; \left\{ \langle ds, a, ds' \rangle \wedge Action\_type(a, type) \right\}$$

$$\langle ds, desc, ds' \rangle \equiv_{def} \exists a \; \left\{ \langle ds, a, ds' \rangle \wedge Turn\_desc(a, desc) \right\}$$

[4]An action execution also has metrical information associated with it. This metrical information represents an estimate of, for example, the distance or the angle between the distinctive states associated with the action execution.

Example 1

Consider a robot moving in the environment depicted in figure 1. The robot moves from distinctive state *a* to distinctive state *b* by performing a follow-midline action, *ml*. Then the robot performs the same action to move to distinctive state *c*. We assume that all corridor intersections look alike ($v+$). This set of experiences can be described by the formulae:

$$Action\_type(ml, travel)\ , CS(s1, a, ml, b)\ ,\ CS(s2, b, ml, c)\ ,$$
$$View(a, v+)\ ,\ View(b, v+)\ ,\ View(c, v+)\ .$$

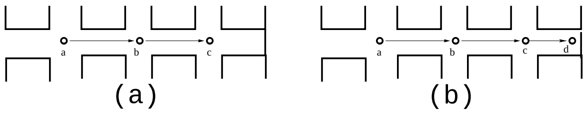

Figure 1: (a) Distinctive states *a*, *b* and *c* are not distinguishable at the causal level. Topological information is needed in order to distinguish them. (b) All distinctive states are distinguished at the causal level given the new information $\langle c, travel, d\rangle$.

Given this set of experiences, at the SSH causal level distinctive states *a*, *b* and *c* are not distinguishable. Any known sequence of actions renders the same set of views. However, at the SSH topological level all these distinctive states are distinguishable since the robot has traveled from *a* to *b* and then to *c* following the same *path* (see example 3). Should the robot continue the exploration and visit distinctive state d, with view $\sqsupset$, then by relying just on known actions and views the agent can distinguish all distinctive states it has visited (see example 2). {*end of example*}

The agent's experiences in the environment are described in terms of *CS*, *View*, *Action_type* and *Turn_desc* atomic formulae. Hereafter we use **E** to denote a particular agent's experience formulae. By $\mathbf{HS}(\mathbf{E})$ we denote the formulae stating that the sorts of schemas, distinctive states, views and actions are completely defined by the sets of *schema*, *distinctive states*, *view* and *action* constant symbols occurring in *E* respectively.[5] By $\mathbf{DT}$ we denote our domain theory, the formulae stating that: (-) the sets {turn, travel}, {turnLeft,turnRight,turnAround}, completely define the sorts of *action_types* and *turn_descriptions*; (-) an action has associated a unique action type ; (-) distinctive states have associated a unique view; (-) the description associated with an action is unique; (-) turn actions have associated a turn description; (-) the type of actions as well as the qualitative description of turn actions is the one specified in *E*. The SSH causal theory **CT(E)** defines when two distinctive states are indistinguishable at the SSH causal level. We use the predicate $ceq(ds, ds')$ to denote this fact. We will assume that actions are *deterministic*: [6]

$$\langle ds, a, ds'\rangle \wedge \langle ds, a, ds''\rangle \rightarrow ds' = ds''\ . \qquad (1)$$

[5]That $sort$ is completely defined by the constant symbols $s_1, \ldots, s_n$ means that an interpretation for $sort$ is the *Herbrand* interpretation defined by the set $\{s_1, \ldots, s_n\}$.

[6]Throughout this paper we assume that formulas are universally quantified.

CT(E) is the following nested abnormality theory [Lifschitz, 1995]:[7]

$$CT(E) \;=\; E,\; HS(E),\; DT,\; Axiom\,1,\; CEQ\_block$$

where **CEQ_block** is defined as

$$\begin{array}{ll} \{\; max\;\; ceq : & \\ \quad ceq(ds, ds') \rightarrow ceq(ds', ds), & \\ \quad ceq(ds, ds') \wedge ceq(ds', ds'') \rightarrow ceq(ds, ds''), & \\ \quad ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v), & (2) \\ \quad ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2') & (3) \\ \} & \end{array}$$

It can be proved that the predicate *ceq* defines an equivalence relation on the sort of distinctive states. Axiom 44 states that indistinguishable distinctive states have the same view. Axiom 44 states that if distinctive states ds and ds' are indistinguishable and action *a* has been performed for both ds and ds', then the action links these states with indistinguishable states. By maximizing ceq we identify distinctive states that cannot be distinguished by actions and/or views, and thereby minimize the set of states represented by the model.[8]

Axioms 44 and 44 allow us to prove the following useful lemma:

Lemma 1 *Let A denote a sequence of action symbols. Let $A(ds)$ denote the distinctive state symbol resulting of starting the sequence A at distinctive state ds or $\perp$ if A is not defined for ds.[9] Then,*

$$\begin{array}{l} ceq(ds, ds') \wedge A(ds) \neq \perp \wedge A(ds') \neq \perp \\ \qquad \rightarrow View(A(ds), v) \equiv View(A(ds'), v) \;\; . \end{array}$$

Example 2

Consider the situation depicted in Figure 1b, with the corresponding schemas and views as in example 1. Using lemma 1 one can conclude that all distinctive states *a*, *b* and *c* are distinguishable by actions and views alone. For instance, $\{ml, ml\}(a) = c$, $\{ml, ml\}(b) = d$, $View(\{ml, ml\}(a), v+)$, $View(\{ml, ml\}(b), \sqsupset)$, and consequently, $\neg ceq(a, b)$. {*end of example*}

The Herbrand models of $CT(E)$ are in a one to one correspondence with the answer sets [Gelfond and Lifschitz, 1991] of the logic program in Figure 2.[10] In this program, the X and Y variables range over distinctive states and the variable V ranges over views in E. The sets of rules 4 and 5 are the facts corresponding to the agent's

[7]See appendix A, page 15.

[8]See appendix D, page 22, for a throughly discussion on the formal properties of ceq.

[9]Given an action symbol A and distinctive state ds, $A(ds) = ds'$ if the schema $\langle ds, A, ds' \rangle$ has been observed, otherwise, $A(ds) = \perp$. Moreover, $A(\perp) = \perp$. The definition is then extended to action sequences in the standard way. Notice that $A(ds)$ being well-defined relies on our assumption that actions are deterministic (Axiom 1).

[10]See appendix C, page 20, for the definition and properties of answer sets. See appendix F, page 29, for a proof of the correctness of the logic program in Figure 2.

experiences. Rule 6 states that an answer set of the program should be *complete* with respect to ceq. Rules 7-9 require ceq to be an equivalence class. Rules 9 and 10 are the counterpart of axiom 44. Rule 12 is the counterpart of axiom 44. In order to define the maximality condition of ceq, the auxiliar predicate $p(X, Y, X1, Y1)$ is introduced. This predicate reads as *"If X and Y were the same, then $X1$ and $Y1$ would be the same"*. The predicate $dist(X, Y)$ defines when distinctive states X and Y are distinguishable. Constraint 13 establishes the maximality condition on ceq: $ceq(X, Y)$ should be the case unless X and Y are distinguishable.[11]

$$\{cs(ds, a, ds') \leftarrow . \quad : cs(ds, a, ds') \in E\} \tag{4}$$
$$\{view(ds, v) \leftarrow . \quad : view(ds, v) \in E\} \tag{5}$$
$$ceq(X, Y), \neg ceq(X, Y) \leftarrow \ .$$
$$p(X, Y, X, Y) \leftarrow .$$
$$p(X, Y, X2, Y1) \leftarrow p(X, Y, X1, Y1), ceq(X1, X2).$$
$$p(X, Y, X1, Y2) \leftarrow p(X, Y, X1, Y1), ceq(Y1, Y2).$$
$$p(X, Y, X2, Y2) \leftarrow p(X, Y, X1, Y1), cs(X1, A, X2), cs(Y1, A, Y2).$$
$$p(X, Y, Y1, X1) \leftarrow p(X, Y, X1, Y1).$$
$$p(X, Y, X1, Y2) \leftarrow p(X, Y, X1, Y1), p(X, Y, Y1, Y2).$$

$$dist(X, Y) \leftarrow p(X, Y, X1, Y1), view(X1, V), not\ view(Y1, V).$$
$$dist(X, Y) \leftarrow p(X, Y, X1, Y1), not\ view(X1, V), view(Y1, V).$$

$$ceq(X, Y); \neg ceq(X, Y) \leftarrow . \tag{6}$$
$$\leftarrow not\ ceq(X, X). \tag{7}$$
$$\leftarrow ceq(X, Y), not\ ceq(Y, X). \tag{8}$$
$$\leftarrow ceq(X, Y), ceq(Y, Z), not\ ceq(X, Z). \tag{9}$$
$$\leftarrow ceq(X, Y), view(X, V), not\ view(Y, V). \tag{10}$$
$$\leftarrow ceq(X, Y), not\ view(X, V), view(Y, V). \tag{11}$$
$$\leftarrow not\ ceq(X1, Y1), ceq(X, Y), cs(X, A, X1), cs(Y, A, Y1). \tag{12}$$

$$\leftarrow not\ ceq(X, Y), not\ dist(X, Y). \tag{13}$$

Figure 2: Logic program associated with CT(E).

4 SSH Topological Level

We are to define the SSH topological theory, $\mathbf{TT(E)}$, associated with a set of experiences E. The language of this theory is a sorted language with sorts for *places*, *paths*

[11] We have implemented this logic program in Smodels [Niemelä and Simons, 1997]. In the implementation, one has to add variable domain restrictions to the different rules. For example, rule

$$ceq(X, Y), \neg ceq(X, Y) \leftarrow \ .$$

becomes

$$ceq(X, Y), \neg ceq(X, Y) \leftarrow \ dstate(X), dstate(Y)$$

where $dstate$ is our predicate to identify the sort of distinctive states.

and *path directions*.[12] The main purpose of $TT(E)$ is to minimize the set of paths and places consistent with the given experiences E. A place can be a *topological place* (hereafter place) or a *region*. A place is a set of distinctive states linked by turn actions. A region is a set of places. We use the predicates *tplace* and *is_region* to identify these subsorts. A path defines an order relation among places connected by travel with no turn actions. They play the role of streets in a city layout. We use the predicate *tpath* to identify the sort of paths. By minimizing the extent of $tplace$, is_region and $tpath$ we minimize the sort of places and paths respectively.[13] The language of the SSH topological level includes the following other predicates: *teq(ds,ds')* – distinctive states ds and ds' are *topologically* indistinguishable; *at(ds,p)* – distinctive state ds is at place p; *along(ds,pa,dir)* –distinctive state ds is along path pa in direction dir; *OnPath(pa,p)* –place p is on path pa; *PO(pa,dir,p,q)* –place p is before place q when facing direction dir on path pa (PO stands for Path Order).

TT(E), is the following nested abnormality theory:

$$\begin{array}{l} \forall p,\ tplace(p) \equiv \neg is\_region(p)\ , \forall pa,\ tpath(pa)\ , \\ \{min\ is\_region\ : \\ \quad CT(E)\ , T\_block\ , AT\_block\quad \} \end{array} \tag{14}$$

The first line in Axioms 14 says that topological places and regions are the two subsorts of places, and that the predicate $tpath$ represents the sort of paths. The block **CT(E)** is the one defined in the previous section. The block **T_block** defines the predicates $\widehat{turn}$, $\widehat{travel}$, and $\overrightarrow{travel}$ such that $\widehat{turn}$ is the equivalence closure of the schemas $\langle \cdot, turn, \cdot \rangle$; $\widehat{travel}$ and $\overrightarrow{travel}$ are the equivalence and transitive closure of the schemas $\langle \cdot, travel, \cdot \rangle$.

The block **AT_block** (Figure 3) is the heart of our theory.[14] The purpose of this block is to define the extent of the predicates *tpath*, *tplace*, *at*, *along*, *PO* and *teq*, while identifying a minimum set of places and paths that explain E. The block has associated the circumscription policy[15]

$$\textbf{circ}\ tpath \succ along \succ PO \succ OnPath \succ tplace\ \textbf{var}\ \vec{SSHpred}$$

where $\vec{SSHpred}$ stands for the tuple of predicates *at*, *teq*, *travel_eq*, and *turn_eq*.[16] This circumscription policy states (among others) that a minimum set of paths is preferred over a minimum set of places. Next we discuss the axioms in AT_block.

Predicate teq is the equivalence relation defined by axiom 15. $teq(ds, ds')$ is the case whenever ds and ds' cannot be distinguished by views and actions (i.e. $ceq(ds, ds')$) and it is consistent to group ds and ds' into the same place. If we assume that views uniquely identify distinctive states (e.g. $View(ds, V) \wedge View(ds', V) \rightarrow ds = ds'$),

[12]The sort of directions is completely defined by the symbols *pos* and *neg*.

[13]Notice that our logic has sorts for *places* and *paths* but in order to minimize these sorts we have to explicitly have predicates representing them.

[14]Notice that the predicate is_region is not mentioned in the theory of figure 3. In the next section we will add to this theory axioms dealing with regions. For the purpose of this section, the minimization of is_region in conjunction with $\forall p,\ tplace(p) \equiv \neg is\_region(p)$ implies (the default) $\forall p\ tplace(p)$.

[15]The symbol $\succ$ indicates prioritized circumscription (see [Lifschitz, 1994] section 7.2).

[16]Block 20 in Figure 3 states that the predicate $turn\_eq$ corresponds to the relation $\widehat{turn}$ modulo teq. Block 32 defines $travel\_eq$ to be the relation $\widehat{travel}$ modulo teq.

$$\{ :$$

$$teq(ds, ds') \equiv \exists p \left\{ ceq(ds, ds') \wedge at(ds, p) \wedge at(ds', p) \right\}, \tag{15}$$

$$at(ds, p) \rightarrow tplace(p), \tag{16}$$

$$\exists! p \, at(ds, p), \tag{17}$$

$$\langle ds, turn, ds' \rangle \wedge at(ds, p) \rightarrow at(ds', p), \tag{18}$$

$$at(ds, p) \wedge at(ds', p) \rightarrow turn\_eq(ds, ds'), \tag{19}$$

$$\begin{aligned} &\{min \;\; turn\_eq : \\ &\quad teq(ds, ds') \wedge teq(dr, dr') \wedge \widehat{turn}(ds', dr') \rightarrow turn\_eq(ds, dr), \\ &\quad turn\_eq(ds, ds') \wedge turn\_eq(ds', ds'') \rightarrow turn\_eq(ds, ds'') \quad \} \end{aligned} \tag{20}$$

$$along(ds, pa, dir) \rightarrow tpath(pa), \tag{21}$$

$$\begin{aligned} &at(ds, p) \wedge at(ds', q) \wedge \vec{travel}(ds, ds') \rightarrow \\ &\quad \exists pa, \; dir \left\{ PO(pa, dir, p, q) \wedge along(ds, pa, dir) \wedge along(ds', pa, dir) \right\}, \end{aligned} \tag{22}$$

$$along(ds, pa, dir) \wedge along(ds, pa1, dir1) \rightarrow pa = pa1, \tag{23}$$

$$\begin{aligned} &at(ds, p) \wedge at(ds', p) \wedge along(ds, pa, dir) \wedge \\ &\quad along(ds', pa, dir) \rightarrow teq(ds, ds'), \end{aligned} \tag{24}$$

$$\begin{aligned} &\Big\{ \langle ds, turn\_desc, ds' \rangle \;\wedge\; turn\_desc \neq turnAround \wedge \\ &\qquad along(ds, pa, dir) \wedge along(ds', pa1, dir1) \Big\} \rightarrow pa \neq pa1, \end{aligned} \tag{25}$$

$$\langle ds, turnAround, ds' \rangle \rightarrow along(ds, pa, dir) \equiv along(ds', pa, -dir), \tag{26}$$

$$PO(pa, pos, p, q) \equiv PO(pa, neg, q, p), \tag{27}$$

$$\neg PO(pa, dir, p, p), \tag{28}$$

$$PO(pa, dir, p, q) \wedge PO(pa, dir, q, r) \rightarrow PO(pa, dir, p, r), \tag{29}$$

$$PO(pa, dir, p, q) \rightarrow OnPath(pa, p) \tag{30}$$

$$\begin{aligned} &OnPath(pa, p) \wedge OnPath(pa, q) \wedge tpath(pa) \rightarrow \\ &\qquad \exists ds, ds' \, \{at(ds, p) \wedge at(ds', q) \wedge travel\_eq(ds, ds')\}, \end{aligned} \tag{31}$$

$$\begin{aligned} &\{min \;\; travel\_eq : \\ &\quad teq(ds, ds') \wedge teq(dr, dr') \wedge \widehat{travel}(ds', dr') \rightarrow travel\_eq(ds, dr), \\ &\quad travel\_eq(ds, ds') \wedge travel\_eq(ds', ds'') \rightarrow travel\_eq(ds, ds'') \;\; \} \end{aligned} \tag{32}$$

$$\textbf{circ} \;\; tpath \succ along \succ PO \succ OnPath \succ tplace \;\; \textbf{var} \;\; SSH\vec{p}red$$

$$\}$$

Figure 3: AT_block.

then predicates *ceq* and *teq* will reduce to equality. This is expected since all that is required to identify a distinctive state is its view.[17]

Every distinctive state is at a unique place (Axiom 17). Whenever the agent *turns*, it stays at the same place (Axiom 18). Distinctive states grouped into a topological place should be *turn* connected (modulo *teq*) (Axiom 19). *Travel* actions among distinctive states are abstracted to topological paths connecting the places associated with those distinctive states (Axiom 22). A distinctive state is along at most one path (Axiom 23). At each place there is at most one distinctive state along a given path direction

[17]See appendix E, page 27, for a discussion of other simplification that can be made when views uniquely identify distinctive states.

(Axiom 24). Turn actions other than *turnAround* change the path the initial and final distinctive states are at (Axiom 25). *TurnAround* actions relate distinctive states being in the same path but opposite directions (Axiom 26). The order of places in a given path direction is the inverse of the order of places in the other path direction (Axiom 27). Axioms 28 and 29 require $PO(pa, dir, \cdot, \cdot)$ to be a non-reflexive transitive order for the places on pa. Places ordered by a path should belong to that path (Axiom 30). Axiom 31 requires the agent to have traveled among the places on a same path.

Our theory does not assume a "rectilinear" environment where paths intersect at most in one place. It is possible for different paths to have the same order of places (see Figure 4). Topological information can distinguish distinctive states not distinguishable by view and actions.

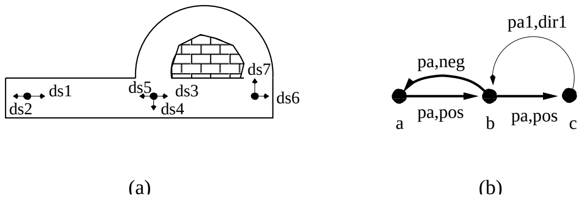

Figure 4: The environment in (a) illustrates a case where different paths intersect at more than one place. (b) depicts the topological map associated with this environment.

Example 3

Consider the scenario of example 1. Since the same view is experienced at *a*, *b* and *c*, the extent of ceq is maximized by declaring $ceq = true$. Using the topological theory, from axiom 17 we conclude that there exist places P and Q, such that $at(a, P)$ and $at(c, Q)$. Since it is the case that $\vec{travel}(a, c)$, from axioms 22 and 28 we conclude, for instance, that $P \neq Q$. Distinctive states a and c are topologically distinguishable though they are "causally indistinguishable" (i.e. $ceq(a, c) \wedge \neg teq(a, c)$). $\{$*end of example*$\}$

Given a minimal model M of $TT(E)$, the SSH topological map is defined by the extent in M of *tpath, tplace, along, PO* and *at*. Since the positive and negative direction of a path are chosen arbitrarily (Axiom 22), there is not a unique minimal model for $TT(E)$. We will consider these "up to path direction isomorphic" models to be the same. However, it is still the case that the theory $TT(E)$ has minimal models that are not isomorphic up to path direction (see Figure 5).

5 SSH Boundary Regions

In addition to connectivity and order among places and paths, the topological map includes topological boundary relations: assertions that a place lies to the right of, to the left of, or on a path. In order to determine boundary relations we formally state the following default heuristic. Suppose the agent is at an intersection on a given path,

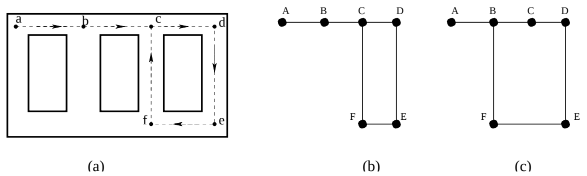

Figure 5: (a) The robot goes around the block visiting places $A,\ldots,F,C$ in the order suggested in the figure. Intersections B and C look alike to the agent. Two minimal models can be associated with the set of experiences in (a) (see (b) and (c)). Topological information is not enough to decide whether the agent is back to B or C. Notice that if the agent accumulates more information, by turning at c and traveling to d, then it can deduce that the topology of the environment is the one in (b). In addition, when available, metrical information can be used to refute the incorrect topology.

and it then turns right. If the agent now travels, any place it finds while traveling with no turns will be on the right of the starting path. When conflicting information exists about whether a place is to the right or left of a path, we deduce no boundary relation (see Figure 6).

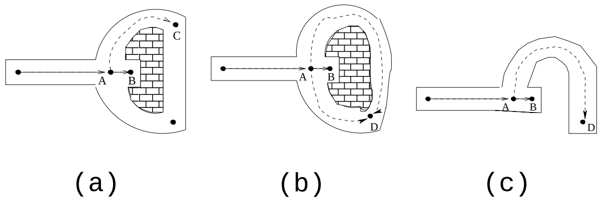

Figure 6: Different environments illustrating how our default to determine boundary relations works. In (a), we conclude by default that place *C* is to the left of the path from *A* to *B*. In (b) we conclude nothing about the location of place *D* with respect to the path from *A* to *B*. In (c), we conclude that place *D* is to the left of the path from *A* to *B*. This is the case since there is no information to conclude otherwise.

We use the predicates $TotheRightOf/TotheLeftOf(p1, pa, dir, pa1, dir1)$ to represent the facts that (i) *p1* is a *place* on both paths, *pa* and *pa1*, and (ii) when the agent is at *place p1* facing in the direction *dir* of *pa*, after executing a turn right (left) action, the agent will be facing on the direction *dir1* of *pa1* (see Figure 7). The predicates *TotheLeftOf* and *TotheRightOf* are derived from the actions performed by the agent at a place:

$$\langle ds, turnRight, ds1\rangle \wedge at(ds, p) \wedge along(ds, pa, dir) \wedge along(ds1, pa1, dir1) \rightarrow TotheRightOf(p, pa, dir, pa1, dir1) \quad (33)$$

We use the predicates $LeftOf(pa, dir, lr)$ and $RightOf(pa, dir, rr)$ to denote that *region* lr (rr) is the left (right) region of path *pa* with respect to the path's direction *dir*. The left/right regions of a path are unique, disjoint, and related when changing the path direction (i.e $LeftOf(pa, dir, r) \equiv RightOf(pa, -dir, r)$). From the relative orientation between paths at a place, we deduce the relative location of places with respect to a path (see Figure 7): [18]

$$TotheRightOf(p1, pa, dir, pa1, dir1) \wedge PO(pa1, dir1, p1, p) \wedge \\ RightOf(pa, dir, rr) \wedge \neg Ab(pa, p) \rightarrow in\_region(p, rr) \qquad (34)$$

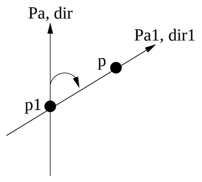

Figure 7: Path *Pa1* is to the right of path *Pa* at place *p1*. Place *p* is after place *p1* on path *pa1*. By default, we conclude that place *p* is to the right of path *pa*.

The predicate **Ab** is the standard "abnormality" predicate used to represent defaults in circumscriptive theories [Lifschitz, 1994]. Axiom 34 states that *"normally"*, if at place *p1* path *pa1* is to the right of path *pa*, and place *p* is after *p1* on path *pa1*, then it should be the case that *p* is on the right of *pa* (Figure 7). In order to capture this default, boundary regions domain theory axioms[19] are added to the block **AT_block** (see Figure 3). Since we are interested in the extent of the new predicates *in_region*, *LeftOf*, *RightOf*, *TotheLeftOf* and *TotheRightOf*, we allow them to vary in the circumscription policy. The new circumscription policy becomes

$$\textbf{circ}\ tpath \succ along \succ PO \succ Onpath \succ \textbf{Ab} \succ \textbf{is\_region} \succ \\ \textbf{in\_region} \succ tplace\ \ \textbf{var}\ \ new\vec{SSH}pred$$

where $new\vec{SSH}pred$ stands for the tuple of predicates *at*, *along*, *teq* , *travel_eq*, *turn_eq*, **LeftOf**, **RightOf**, **TotheLeftOf**, and **TotheRightOf**. The circumscription policy states that boundary relations should be established even at the expense of having more places on the map. In addition, by minimizing the predicates *is_region* and *in_region*, we require the models of our theory to have only the regions that are explicitly created by the agent, and not arbitrary ones.

Example 4

Boundary relations determine distinctions among environment states that could not be derived from the connectivity of places alone. Consider an agent visiting the different corners of a square room in the order suggested by Figure 8a. In addition, suppose the agent defines *views* by characterizing the direction of walls and open space. Accordingly, the agent experiences *four* different views, *v1-v4*, in this environment.

The set of experiences *E* in the environment are:

$$\begin{array}{lll} View(ds1, v1) & View(ds2, v2) & View(ds3, v1) \\ View(ds4, v2) & View(ds5, v1) & \langle ds1, turnRight, ds2\rangle \\ \langle ds2, travel, ds3\rangle & \langle ds4, travel, ds5\rangle & \langle ds3, turnRight, ds4\rangle \end{array}$$

[18]The predicate *in_region(p,r)* states that *place* p is in *region* r.

[19]In the spirit of axioms 33-34.

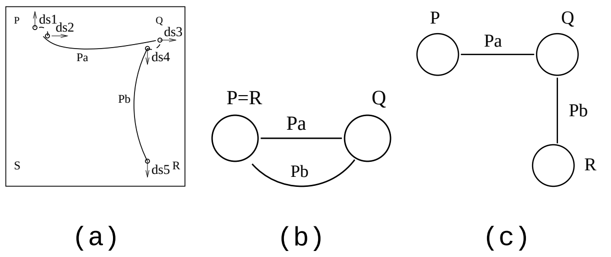

Figure 8: (a) The figure shows the sequence of actions followed by an agent while navigating a square room. Starting at distinctive state ds1, distinctive states are visited in the order suggested by their number. Dashed lines indicate Turn actions. Solid lines indicate Travel actions. (b) and (c) depict the topological map associated with the environment in (a) without and using boundary regions, respectively.

Suppose that the agent does not use boundary regions when building the topological map. Then the minimal topological model associated with *E* has two paths[20] and two places. In this model, $teq(ds1, ds5)$ is the case. The environment looks perfectly symmetric to the agent (Figure 8b).!!

Suppose now that the agent relies on boundary regions. Let *P*, *Q*, *R*, be the topological places associated with *ds1*, *ds3* and *ds5* respectively. From Axiom 22, let *Pa*, *Pb*, dir_a and dir_b be such that $PO(Pa, dir_a, P, Q)$, $along(ds2, Pa, dir_a)$, $along(ds3, Pa, dir_a)$, $PO(Pb, dir_b, Q, R)$, $along(ds4, Pb, dir_b)$, and $along(ds5, Pb, dir_b)$ hold. From Axiom 33 we can conclude then $TotheRightOf(Q, Pa, dir_a, Pb, dir_b)$. In the proposed model, the extent of *Ab* is minimized by declaring $Ab = false$ and consequently from Axiom 34 we conclude $in\_region(R, right(Pa, dir_a))$ where $right(Pa, dir_a)$ denotes the right region of Pa when facing dir_a. Finally, since a path and its regions are disjoint, and $OnPath(Pa, P)$ is the case, we conclude $P \neq R$ and so $\neq teq(ds1, ds5)$. The resulting topological map is depicted in Figure 8c. {*end of example*}

If the agent's sensory capabilities are so impoverished that many distinctive states are perceived to be similar, then metrical information could be used to distinguish different environment states. Figure 9 summarizes different representations an agent could build depending on the spatial properties it relies on.

6 Conclusions

Starting with an informal description of the SSH we have formally specified its intended models. These models correspond to the models of the circumscriptive theory *TT(E)*. The formal account of the theory allows us to illustrate the deductive power of the different SSH ontologies. For instance, example 4 shows how the use of boundary relations allows the agent to determine distinctions among environment states that could not be derived from the connectivity of places and paths alone.

[20]Notice that from $\langle ds3, turnRight, ds4\rangle$ and Axiom 25 we can deduce that $Pa \neq Pb$ in Figure 8b.

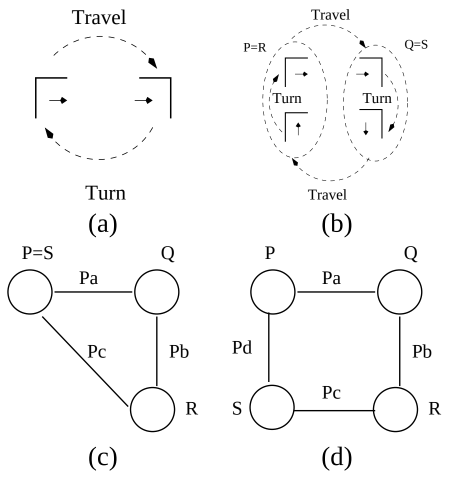

Figure 9: Consider the same environment and agent as in figure 8. Assumes the agent keeps turning right and following the left wall until it is back to distinctive state $ds1$, at place P. Only two kind of views $|\rightrightarrows$ and $\rightrightarrows|$ are observed by the agent. Next we summarizes different maps the agent could build depending on the spatial properties it relies on. (a) If the agent only relies on causal information, the map consists of two states. (b) When topological information is used, but without boundary relations, the map consists of four states and two places. (c) When boundary relations are used, the map consists of six states and three places. There is no fixed correspondence between the three places in the map and the four indistinguishable places in the real world. (d) If metrical information is accurate enough to refute the hypothesis $P = S$, the map will consist of eight states and four places.

The theory *TT(E)* is rather complex so it may be difficult to determine the effect of the different defaults in combination. However, it is possible to translate this theory into a logic program whose answer sets determine the models of *TT(E)*. We have illustrated the case for the SSH causal theory *CT(E)*, but the same techniques apply for *TT(E)*. The major subtleties in the translation are the minimality and maximality conditions associated with the theory. We have used Smodels to calculate the models of *TT(E)* and confirm that the theory yields the intended models. However, when the number of distinctive states is big, Smodels may not be able to ground the theory as the number of rules associated with the program grows exponentially. We are still working on solving this problem.

Acknowledgments

We are grateful to Vladimir Lifschitz for his valuable feedback during this work and for suggesting the use of Smodels to implement the ideas proposed here.

A Nested Abnormality theories

In this appendix we define circumscription and nested abnormalities theories following [Lifschitz, 1994, Lifschitz, 1995].

The main idea of circumscription is to consider, instead of arbitrary models of an axiom set, only the models that satisfy a certain minimality condition (usually set inclusion). Mathematically, circumscription is defined as a syntactic transformation of logical formulas. It transforms a sentence A into a stronger sentence A^*, such that the models of A^* are precisely the minimal models of A.

Definition 1 (Circumscription)

Let $A(P, Z_1, \ldots, Z_m)$ be a sentence containing a predicate constant P and object, function and/or predicate constants $Z_1, \ldots, Z_m$ (and possibly other object, function and predicate constants). The *circumscription of P in A with varied* $Z_1, \ldots, Z_m$ is the sentence

$$A(P, Z_1, \ldots, Z_m) \wedge \neg\exists p, z_1, \ldots, z_m \, [A(p, z_1, \ldots, z_m) \wedge p < P] \qquad (35)$$

where $p < P$ denotes the formula

$$\forall x \; \{p(x) \rightarrow P(x)\} \; \wedge \exists x \, \{p(x) \wedge \neg P(x)\} \quad .$$

We denote the formula 35 by $CIRC\,[A; P; Z]$. $\{$*end of definition*$\}$

Intuitively, the models of $CIRC\,[A; P; Z]$ are the models of A in which the extent of P cannot be smaller without losing the property A, even at the price of changing the interpretations of the constants Z. In order to make this claim precisely, the following order, $\leq^{P;Z}$, is defined among structures of the language of A.

Definition 2 ($\leq^{P;Z}$)

Let M_1 and M_2 be two structures for a given one-sorted language. Then $M_1 \leq^{P;Z} M_2$ whenever

- $|M_1| = |M_2|$,[21]
- $M_1\,[C] = M_2\,[C]$,[22] for every constant C which is different from P and does not belong to Z,
- $M_1\,[P] \subseteq M_2\,[P]$

$\{$*end of definition*$\}$

$M_1 \leq^{P;Z} M_2$ means that M_1 and M_2 differ only in how they interpret P and Z, and the extent of P in M_1 is a subset of its extent in M_2. The next relation (proposition 1 in [Lifschitz, 1994]) relates $CIRC\,[A; P; Z]$ and $\leq^{P;Z}$:

[21]For a structure M, $|M|$ denotes the universe of M.
[22]$M\,[C]$ denotes the interpretation of C in M.

Theorem 1 *A structure M is a model of $CIRC\,[A; P; Z]$ if and only if M is minimal relative to $\leq^{P;Z}$.*

Example 5

Circumscription is usually used in order to formalize default associated with an axiomatic theory. Suppose the we would like to represent the default "Normally a block in on the table". Suppose block B_1 is not on the table and let B_2 denotes another block. The circumscriptive theory below allow us to conclude $Ontable(B_2)$:

$$Block(x) \wedge \neg Ab(x) \rightarrow Ontable(x) \tag{36}$$

$$\neg Ontable(B_1) \tag{37}$$

$$Block(B_1),\ Block(B_2),\ \ B_1 \neq B_2 \tag{38}$$

$$\textbf{circ}\ \ Ab\ \ \textbf{var}\ \ Ontable \tag{39}$$

where we have extended our notation such that the above should be understood as

$$CIRC\,[36 \wedge 37 \wedge 38; Ab; Ontable] \quad .$$

In the above theory, it is the case that $Ab(x) \equiv x = B_1$, and consequently $Ontable(B_2)$ follows. The role of the predicate Ab is to single out the blocks that are *"abnormal"* relative to the default (36). $\{$*end of example*$\}$

It is often convenient to arrange different defaults by assigning priorities to them. For example, consider formalizing the enhancement of the theory above in which "Blocks are usually heavy", and "heavy block are usually not in the table". Next we define two extensions two the basic definition of circumscription: parallel and prioritized circumscription.

Definition 3 (Parallel Circumscription)

The *parallel circumscription*

$$CIRC\left[A; P^1, \ldots, P^k; Z\right]$$

is the sentence

$$A(P, Z) \wedge \neg \exists p, z\,[A(p, z) \wedge p \prec P] \quad ,$$

where P stands for the tuple of predicates $P^1, \ldots, P^n$ and $p \prec P$ stands for the formula $\forall\ 1 \leq i \leq n\ p^i \leq P^i\ \ \wedge \exists\ 1 \leq i \leq n\ p^i < P^i$
$\{$*end of definition*$\}$

The parallel circumscription of several predicates has a simple model theoretics characterization, similar to the one presented by theorem 1. When P is a tuple $P^1, \ldots, P^n$, the relation $M_1 \leq^{P;Z} M_2$ between structures M_1 and M_2 is defined as before, except that the condition $M_1\,[P] \subseteq M_2\,[P]$ is replaced by $M_1\left[P^i\right] \subseteq M_2\left[P^i\right]$ for all $i = 1, \ldots, n$.

Definition 4 (Prioritized Circumscription)

The *prioritized circumscription*

$$CIRC\left[A; P^1 \succ \ldots \succ P^k; Z\right]$$

is the sentence

$$A(P,Z) \wedge \neg\exists p, z\,[A(p,z) \wedge p \prec P] \quad ,$$

where P stands for the tuple of predicates $P^1, \ldots, P^n$ and $p \prec P$ stands for the formula

$$\bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{i-1} (p^j = P^j) \wedge (p^i < P^i) \right) \quad .$$

$\{$*end of definition*$\}$

The formula $p \prec P$ defines a *lexicographic* order among the predicates in p and P. When $k = 1$ it becomes $p < P$; if $k = 2$, it becomes

$$(p^1 < P^1) \vee ((p^1 = P^1) \rightarrow (p^2 < P^2)) \quad .$$

Proposition 15 in [Lifschitz, 1994] shows that prioritized circumscription can be reduced to parallel circumscription as follows:

Theorem 2 *The circumscription $CIRC\left[A; P^1 \succ \ldots \succ P^k; Z\right]$ is equivalent to*

$$\bigwedge_{i=1}^{k} CIRC\left[A; P^i; P^{i+1}, \ldots, P^k, Z\right] \quad .$$

B Nested Abnormality theories (NAT's)

Nested abnormality theories allows one to apply the circumscription operator to a subset of axioms, by structuring the knowledge base (the theory) into blocks. Each block can be viewed as a group of axioms that describes a certain collection of predicates and functions, and the nesting of blocks reflects the dependence of these descriptions on each other.

Definition 5 (NAT's)

Consider a second-order language L that does *not* include Ab among its symbols. For every natural number k, by L_k we denote the language obtained from L by adding Ab as a k-ary predicate constant. *Blocks* are defined recursively as follows: For any k and any list of function and/or predicate constants $C_1, \ldots, C_m$ of L, if each of $A_1, \ldots, A_n$ is a formula of L_k or a *block*, then $\{C_1, \ldots, C_m \,:\, A_1, \ldots, A_n\}$ is a *block*. The last expression reads: $C_1, \ldots, C_m$ are such that $A_1, \ldots, A_n$. About $C_1, \ldots, C_m$ we say that they are *described* by this block.

The semantics of NAT's is characterized by a map φ that translates blocks into sentences of L. It is convenient to make φ defined also on formulas of the languages L_k. If A is such a formula, then $\varphi(A)$ stands for the universal closure of A. For blocks we define, recursively:

$$\varphi\{C_1,\ldots,C_m \ :\ A_1,\ldots,A_n\} = \exists ab\, CIRC\,[\varphi A_1,\ldots,\varphi A_n \ :\ ab \ :\ C_1,\ldots,C_m] \quad .$$

{*end of definition*}

Example 6

Consider the standard example: objects normally don't fly: birds normally do; canaries are birds; Tweety is a canary. These assertions can be formalized as the NAT whose only axiom is

$$\begin{array}{l} \{Flies\ : \\ \quad Flies(x) \rightarrow Ab(x), \\ \quad \{\ Flies\ : \\ \quad\ \ Bird(x) \wedge \neg Ab(x) \rightarrow Flies(x), \\ \quad\ \ Canary(x) \rightarrow Bird(x), \\ \quad\ \ Canary(Tweety) \\ \quad \} \\ \} \end{array}$$

The outer block describe the ability of objects to fly; the inner block gives more specific information about the ability of *birds* to fly. Each occurrence of the predicate Ab is "local" to its block, and so, the two occurrences of the predicate Ab refer two "unrelated" predicates though we use the same name.
{*end of example*}

Most often, it is desirable not to mention the predicate Ab at all. We will adopt the following notations:

- $\{C_1,\ldots,C_m, min\ P \ :\ A_1,\ldots,A_n\}$ stands for

$$\{C_1,\ldots,C_m, P \ :\ P(x) \rightarrow Ab(x),\ A_1,\ldots,A_n\}$$

- $\{C_1,\ldots,C_m, max\ P \ :\ A_1,\ldots,A_n\}$ stands for

$$\{C_1,\ldots,C_m, P \ :\ \neg Ab(x) \rightarrow P(x),\ A_1,\ldots,A_n\}$$

Using this notation, we could rewrite our previous example as

$$\begin{array}{l} \{min\ Flies\ : \\ \quad \{\ Flies\ : \end{array}$$

$$
\begin{aligned}
&Bird(x) \wedge \neg Ab(x) \rightarrow Flies(x),\\
&Canary(x) \rightarrow Bird(x),\\
&Canary(Tweety)\\
&\}\\
\}
\end{aligned}
$$

where we dispense the occurrence of one Ab predicate. The reader is referred to [McCarthy, 1980, McCarthy, 1986, Lifschitz, 1994, Lifschitz, 1995] for a complete survey of the uses and properties of circumscription and NATs.

C Answer Sets

In this appendix we defined the answer set semantics for a logic program as defined in [Lifschitz, 1999, Gelfond and Lifschitz, 1991].

Consider a set of propositional symbols, called **atoms**. A **literal** is an expression of the form A or $\neg A$, where A is an atom (we call the symbol $\neg$ "classical negation", to distinguish it from the symbol not used for negation as failure). A rule element is an expression of the form L or $not\, L$, where L is a literal. A **rule** is an ordered pair

$$Head \leftarrow Body \tag{40}$$

where $Head$ and $Body$ are finite sets of rule elements. If

$$Head = \{L_1, \ldots, L_k, not\, L_{k+1}, \ldots, not\, L_l\}$$

and

$$Body = \{L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n\}$$

($n \geq m \geq l \geq k \geq 0$) then we write (40) as

$$L_1; \ldots, L_k; not\, L_{k+1}, \ldots, not\, L_l \leftarrow L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \quad .$$

A rule (40) is a **constraint** if $Head = \emptyset$. A program is a set of rules.

The notion of an answer set is defined first for program that do not contain negation as failure ($l = k$ and $n = m$ in every rule of the program). Let Π be such program, and let X be a consistent set of literals. We say that X is **closed** under Π if, for every rule in Π, $Head \cap X \neq \emptyset$ whenever $Body \subseteq X$. We say that X is an **answer set** of Π if X is minimal among the sets closed under Π (relative to set inclusion).

Example 7

The program

$$p; q \leftarrow$$
$$\neg r \leftarrow p$$

has two answer sets: $\{p, \neg r\}$ and $\{q\}$. If we add the constraint

$$\leftarrow q$$

to this program, we will get a program whose only answer set is $\{p, \neg r\}$ (see theorem 3 below). $\{$*end of example*$\}$

To extend the definition of an answer set to programs with negation as failure, take any program Π, and let X be a consistent set of literals. The **reduct** Π^X of Π relative to X is the set of rules

$$L_1; \ldots, L_k; \leftarrow L_{l+1}, \ldots, L_m \quad ,$$

for all rules (40) in Π such that X contains all the literals $L_{k+1}, \ldots, L_l$ but does not contain any of the $L_{m+1}, \ldots, L_n$. Thus Π^X is a program without negation as failure. We say that X is an **answer set** for Π if X is an answer set for Π^X.

Example 8

The program

$$p \leftarrow not\, q$$
$$q \leftarrow not\, p$$

has two answer sets: $\{p\}$ and $\{q\}$. $\{$*end of example*$\}$

Adding a constraint to a program affects its collection of answer sets by eliminating the answer sets that "violate" this constraint. Next we prove this property of answer sets.

Theorem 3 *Let Π_1 and Π_2 be logic programs such that Π_2 is obtained from Π_1 by adding a set of constraints C (i.e. $\Pi_2 = \Pi_1 \cup C$). Let X be a consistent set of literals. Then X is an answer set for Π_2 if and only if X is an answer set for Π_1 such that for each rule $\leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \ \in C$, $\{L_1, \ldots, L_m\} \not\subseteq X$ whenever X does not contain any of $L_{m+1}, \ldots, L_n$.*

Proof. Let X and Y be consistent sets of literals. Let $C^X(Y)$ denote the fact that Y does not violates any constraint in C^X, that is, if $\leftarrow L_1, \ldots, L_m \ \in C^X$ then $\{L_1, \ldots, L_m\} \not\subseteq Y$. In particular,

$$Y \subseteq X \wedge C^X(X) \rightarrow C^X(Y) \tag{41}$$

Moreover, it is the case that

$$C^X(Y) \equiv Y \ is\ closed\ under\ C^X \tag{42}$$

In fact, if Y is closed under C^X and $\leftarrow L_1, \ldots, L_m \ \in C^X$, then $\{L_1, \ldots, L_m\} \not\subseteq Y$ for otherwise $Y \cap \emptyset \neq \emptyset$. Conversely, if $C^X(Y)$ is the case, it is easy to see that Y is closed under C^X. From the definition of reduct we have that $\Pi_2^X = \Pi_1^X \ cup C^X$. Using (41) it is then the case that

$$\{Y :\ closed_{\Pi_2^X}(Y)\} \ = \ \{Y :\ closed_{\Pi_1^X}(Y) \wedge C^X(Y)\} \tag{43}$$

From facts (41) and (43) the theorem follows:

$$\begin{array}{ll}
\multicolumn{2}{l}{X\ answer\ set\ for\ \Pi_2} \\
\equiv & X\ answer\ set\ for\ \Pi_2^X \\
\equiv & X\ minimal\ of\ \{Y :\ closed_{\Pi_2^X}(Y)\} \\
\overset{(43)}{\equiv} & X\ minimal\ of\ \{Y :\ closed_{\Pi_1^X}(Y) \wedge C^X(Y)\} \\
\equiv & C^X(X) \wedge \forall Y \left(Y \neq X \wedge Y \subseteq X \wedge C^X(Y) \rightarrow \neg closed_{\Pi_1^X}(Y)\right) \\
\overset{(41)}{\equiv} & C^X(X) \wedge \forall Y \left(Y \neq X \wedge Y \subseteq X \rightarrow \neg closed_{\Pi_1^X}(Y)\right) \\
\equiv & C^X(X) \wedge X\ answer\ set\ for\ \Pi_1^X \\
\equiv & C^X(X) \wedge X\ answer\ set\ for\ \Pi_1
\end{array}$$

□

D Ceq properties

In this appendix we provide proofs for the different properties of the predicated ceq defined in section 3 (Page 6). We start by proving that predicate ceq is indeed an equivalence relation.[23] We then illustrate that it in general it is necessary to explicitly ask ceq to be symmetric and transitive.[24] We show that have the agent completely explored the environment, then the maximization principle defining ceq will guaranty that ceq is an equivalence relation without explicitly requiring so (Page 6). Moreover, we show that in this case the predicate ceq captures the idea that two distinctive states are the same if they render the same views under any sequence of actions.

Theorem 4 *The predicate ceq is an equivalence relation.*

Proof. For the purpose of the proof next we reproduce the definition of the theory *CT(E)*:

$$\begin{array}{l}
CT(E)\ = \\
\quad E,\ HS(E),\ DT, \\
\quad \langle ds, a, ds' \rangle \wedge \langle ds, a, ds'' \rangle \rightarrow ds' = ds'', \qquad (Axiom\ 1) \\
\quad CEQ\_block\ = \\
\quad \{\ max\ \ ceq: \\
\quad\ \ ceq(ds, ds') \rightarrow ceq(ds', ds), \\
\quad\ \ ceq(ds, ds') \wedge ceq(ds', ds'') \rightarrow ceq(ds, ds''), \\
\\
\quad\ \ ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v), \\
\quad\ \ ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2') \\
\quad \}
\end{array}$$

We need to prove that $ceq(ds, ds)$ is the case. Let M_1 be a model for the axioms inside the *CEQ_block* as well as the other axioms of $CT(E)$. Let M_2 be a structure identical to M_1 except that

$$ceq^{M_2}(ds, ds') \equiv ceq^{M_1}(ds, ds') \vee ds = ds' \quad .$$

Our theorem follows once we prove that M_2 is a model for the axioms inside the *CEQ_block*.[25] Next we show why this is the case:

- $M_2 \models ceq(ds, ds') \rightarrow ceq(ds', ds)$. In fact,

$$\begin{array}{ll}
\multicolumn{2}{l}{ceq^{M_2}(ds, ds')} \\
\equiv & ceq^{M_1}(ds, ds') \vee ds = ds' \\
\rightarrow & ceq^{M_1}(ds', ds) \vee ds' = ds \\
\equiv & ceq^{M_2}(ds', ds)
\end{array}$$

[23]See ceq's definition below.

[24]The maximization associated with ceq's definition does not guaranty these properties.

[25]M_2 satisfies the other axioms in $CT(E)$ since ceq does not occur in them.

- $M_2 \models ceq(ds, ds') \wedge ceq(ds', ds'') \rightarrow ceq(ds, ds'')$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds, ds') \wedge ceq^{M_2}(ds', ds'')} \\ \equiv & \big(ceq^{M_1}(ds, ds') \vee ds = ds'\big) \wedge \big(ceq^{M_1}(ds', ds'') \vee ds' = ds''\big) \\ \equiv & \big(ceq^{M_1}(ds, ds') \wedge ceq^{M_1}(ds', ds'')\big) \vee \big(ds = ds' \wedge ceq^{M_1}(ds', ds'')\big) \vee \\ & \big(ceq^{M_1}(ds, ds') \wedge ds' = ds''\big) \vee (ds = ds' \wedge ds' = ds'') \\ \rightarrow & ceq^{M_1}(ds, ds'') \vee (ds = ds' \wedge ds' = ds'') \\ \equiv & ceq^{M_2}(ds, ds'') \end{array}$$

- $M_2 \models ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v)$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds, ds')} \\ \equiv & ceq^{M_1}(ds, ds') \vee ds = ds' \\ \rightarrow & \forall v\,[View(ds, v) \equiv View(ds', v)] \vee ds = ds' \\ \rightarrow & \forall v\,[View(ds, v) \equiv View(ds', v)] \vee \forall v\,[View(ds, v) \equiv View(ds', v)] \\ \equiv & View(ds, v) \equiv View(ds', v) \end{array}$$

- $M_2 \models ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2'')$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle} \\ \equiv & \big(ceq^{M_1}(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle\big) \vee \\ & (ds_1 = ds_2 \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle) \\ \rightarrow & ceq^{M_1}(ds_1', ds_2') \vee (\langle ds_1, a, ds_1' \rangle \wedge \langle ds_1, a, ds_2' \rangle) \\ \overset{(1)}{\rightarrow} & ceq^{M_1}(ds_1', ds_2') \vee ds_1' = ds_2' \\ \equiv & ceq^{M_2}(ds_1', ds_2') \end{array}$$

$\square$

Axiom 1 (i.e. actions are deterministic) is fundamental in the proof above. Without this axiom, we could have a set of experiences like

$$\begin{array}{l} Action\_type(ml, travel)\ , \\ CS(s1, a, ml, b)\ ,\ CS(s2, a, ml, c)\ . \\ View(a, v)\ ,\ View(b, v1)\ ,\ View(c, v2) \end{array}$$

for which $ceq(a, a)$ is not the case.

In general it is not possible to remove the ceq's symmetry and transitivity axioms from inside CEQ_block. Consider the following example.

Example 9

Let E be the set defined by the following formulae:

$$\begin{array}{l} CS(s1, ds_1, a_1, ds_2)\ ,\ CS(s2, ds_1, a_2, ds_3)\ , \\ CS(s3, ds_2, a_3, ds_4)\ ,\ CS(s4, ds_3, a_3, ds_5)\ , \\ View(ds_1, v)\ ,\ View(ds_2, v)\ ,\ View(ds_3, v)\ , \\ View(ds_4, v1)\ ,\ View(ds_5, v2)\ . \end{array}$$

Suppose our definition of CEQ_block were :

$$\begin{array}{l} CEQ\_block\ = \\ \quad \{\ max\ \ ceq : \\ \quad\ \ ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v), \\ \quad\ \ ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2') \\ \quad \} \end{array}$$

Since $v1 \neq v2$ we conclude $\neg ceq(ds_4, ds_5)$. This in turn implies $\neg ceq(ds_2, ds3)$. However, in order to maximize *ceq* we can make $ceq(ds_1, ds_2) \wedge ceq(ds_1, ds_3) \wedge ceq(ds_2, ds_1) \wedge ceq(ds_3, ds_1) \wedge \forall ds\, ceq(ds, ds)$ to be the case. In such a model, *ceq* is not transitive.
{*end of example*}

There is a special case in which *ceq* is symmetric and transitive without explicitly tell so. This is the case when the result of every action at every distinctive state is known. In this case, we said that the set of experiences is complete.

Definition 6

A set of experiences E is **complete** whenever

$$E \models \forall a, ds \exists ds' \langle ds, a, ds' \rangle \ \ .$$

{*end of definition*}

Theorem 5

Let E be a complete set of experiences. Let *CT(E)* be defined as follows:

$$\begin{array}{l} CT(E)\ = \\ \quad E,\ HS(E),\ DT, \\ \quad \langle ds, a, ds' \rangle \wedge \langle ds, a, ds'' \rangle \rightarrow ds' = ds'', \qquad (Axiom\ 1) \\ \quad CEQ\_block\ = \\ \quad \{\ max\ \ ceq : \\ \quad\ \ ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v), \\ \quad\ \ ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2') \\ \quad \} \end{array}$$

Then the predicate *ceq* is an equivalence relation.

Proof. Let M_1 be a model for the axioms inside the *CEQ_block* as well as the other axioms of $CT(E)$. We need to prove that it is possible to have a structure M_2 identical to M_1 except that $ceq^{M_1} \subseteq ceq^{M_2}$, M_2 is a model for the axioms inside the *CEQ_block*, the other axioms of $CT(E)$, and ceq^{M_2} is an equivalence class. The proof goes along the lines of theorem's 4 proof. Indeed, the same proof as in theorem 4 allow us to assume that $M1$ is reflexive.

Let M_2 be a model identical to M_1 except that

$$ceq^{M_2}(ds, ds') = ceq^{M_1}(ds, ds') \vee ceq^{M_1}(ds', ds) \ .$$

By definition, ceq^{M_2} is symmetric. We need to prove that M_2 satisfy the axioms inside (the new) CEQ_block:

- $M_2 \models ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v)$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds, ds')} \\ \equiv & ceq^{M_1}(ds, ds') \vee ceq^{M_1}(ds', ds) \\ \rightarrow & \forall v\, [View(ds, v) \equiv View(ds', v)] \vee \forall v\, [View(ds', v) \equiv View(ds, v)] \\ \equiv & View(ds, v) \equiv View(ds', v) \end{array}$$

- $M_2 \models ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds'_1\rangle \wedge \langle ds_2, a, ds'_2\rangle \rightarrow ceq(ds'_1, ds'_2)$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds_1, ds_2) \wedge \langle ds_1, a, ds'_1\rangle \wedge \langle ds_2, a, ds'_2\rangle} \\ \equiv & \left[ceq^{M_1}(ds_1, ds_2) \wedge \langle ds_1, a, ds'_1\rangle \wedge \langle ds_2, a, ds'_2\rangle\right] \vee \\ & \left[ceq^{M_1}(ds_2, ds_1) \wedge \langle ds_1, a, ds'_1\rangle \wedge \langle ds_2, a, ds'_2\rangle\right] \\ \rightarrow & ceq^{M_1}(ds'_1, ds'_2) \vee ceq^{M_1}(ds'_2, ds'_1) \\ \equiv & ceq^{M_2}(ds'_1, ds'_2) \end{array}$$

Finally, let M_2 be a model identical to M_1 except that

$$ceq^{M_2} = transitive\_closure(ceq^{M_1}) \ .$$

By definition, ceq^{M_2} is transitive. If ceq^{M_1} is reflexive and simmetric, so is ceq^{M_2}. We need to prove that M_2 satisfies the axioms inside (the new) CEQ_block:

- $M_2 \models ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v)$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds, ds')} \\ \equiv & \exists ds^0, ds^1, \ldots, ds^n \left[ds = ds^0,\, ds' = ds^n,\, ceq^{M_1}(ds^i, ds^{i+1}),\ 0 \le i < n\right] \\ \rightarrow & \exists ds^0, ds^1, \ldots, ds^n \\ & \qquad \left[ds = ds^0,\, ds' = ds^n,\, View(ds^i, v) \equiv View(ds^{i+1}, v),\ 0 \le i < n\right] \\ \rightarrow & \exists ds^0, ds^n \ \left[ds = ds^0,\, ds' = ds^n, View(ds^0, v) \equiv View(ds^n, v)\right] \\ \equiv & View(ds, v) \equiv View(ds', v) \end{array}$$

- $M_2 \models ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2')$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle} \\ \equiv & \exists ds^i (1 \leq i \leq n) \left[ds_1 = ds^1,\ ds_2 = ds^n,\ ceq^{M_1}(ds^i, ds^{i+1}),\ 1 \leq i < n\right] \\ & \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \\ \overset{hyp.}{\rightarrow} & \exists ds^i \exists \langle ds^i, a, ds^{i'} \rangle \\ & \left[ds_1 = ds^1, ds_2 = ds^n, ds_1' = ds^{1'}, ds_2' = ds^{n'}, ceq^{M_1}(ds^i, ds^{i+1}),\ 1 \leq i < n\right] \\ \rightarrow & \exists ds^{i'} \left[ds_1' = ds^{1'},\ ds_2' = ds^{n'},\ ceq^{M_1}(ds^{i'}, ds^{(i+1)'}),\ 1 \leq i < n\right] \\ \equiv & ceq^{M_2}(ds_1', ds_2') \end{array}$$

□

When a set of experiences is complete the predicate *ceq* captures the idea that two distinctive states are the same if they render the same views under any sequence of actions. Assume that E is complete and let $A^* = a_1, \ldots, a_n$ a sequence of actions. The term $T(A^*)(ds)$ denotes the distinctive state resulting from executing A^* starting at ds. By definition, $T(A^*)(ds) = ds$ if $n = 0$, $T(A*)(ds) = ds'$ such that $(E \models \langle T(a_1, \ldots, a_{n-1})(ds), a_n, ds' \rangle$. Notice that the definition of $T(A^*)(ds)$ makes sense since E is complete and actions are deterministic.

Theorem 6 *Let E be a complete set of experiences. Then,*

$$ceq(ds, ds') \equiv \forall A^*, v\ \ [View(T(A^*)(ds), v) \equiv View(T(A^*)(ds'), v)]\ .$$

Proof. Let M_1 be a model for the axioms inside the *CEQ_block* as well as the other axioms of $CT(E)$. Let M_2 be a model identical to M_1 except that

$$ceq^{M_2}(ds, ds') \equiv \forall A^*, v\ \ [View(T(A^*)(ds), v) \equiv View(T(A^*)(ds'), v)]\ .$$

By induction in the length of action sequences on can prove that $ceq^{M_1} \subseteq ceq^{M_2}$. Our proof is complete by showing that M_2 satisfy the axioms inside (the new) CEQ_block:

- $M_2 \models ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v)$. In fact, suppose $M_2 \models ceq(ds, ds')$ and consider the empty sequence of actions. Then

  $$View(ds, V) \equiv View(T(\{\})(ds), v) \equiv View(T(\{\})(ds'), v) \equiv View(ds', v)\ .$$

- $M_2 \models ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2')$. In fact,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq^{M_2}(ds_1', ds_2')} \\ \equiv & \forall A^*, v\ \left[View(T(A^*)(ds_1'), v) \equiv view(T(A^*)(ds_2'), v)\right] \\ \leftarrow & \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle\ \wedge \\ & \quad \forall A^*, v\ \ [View(T(aA^*)(ds_1), v) \equiv View(T(aA^*)(ds_2), v)] \\ \leftarrow & ceq^{M_2}(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \end{array}$$

□

E What if Views uniquely identify distinctive states

In this section we explore how our theory is simplified by the assumption that views uniquely identify distinctive states. Under this hypothesis, we add the following axiom to our theory:

$$View(ds, V) \wedge View(ds', V) \rightarrow ds = ds' \tag{44}$$

Since $=$ satisfies all the axioms in **CEQ_block** (Page 6), we know that $= \subseteq ceq$. From 44 and 44 we conclude that $ceq \subseteq =$. Consequently, we do not need to have block CEQ_block and we can replace ceq by $=$.

Let's consider the predicate teq defined by Axiom 15 (Page 9). Since ceq is equal to $=$, we deduce that

$$\begin{aligned} teq(ds, ds') &\equiv \exists p \, \{ds = ds' \wedge at(ds, p) \wedge at(ds', p)\} \\ &\equiv ds = ds' \wedge \exists p \; at(ds, p) \\ &\equiv ds = ds' \end{aligned}$$

Consequently, teq can also be replaced by $=$. The fact that ceq and teq reduce to $=$ is expected since all that is required to identify a distinctive state is its view. Next we show that the predicates $turn\_eq$ and $travel\_eq$ can be replaced by $\widehat{turn}$ and $\widehat{travel}$, respectively.

By replacing teq by $=$, block 20 can be rewritten as:

$$\begin{aligned}
\{ \quad & min \;\; turn\_eq : \\
& \widehat{turn}(ds', dr') \wedge teq(ds, ds') \wedge teq(dr, dr') \rightarrow turn\_eq(ds, dr) \\
& turn\_eq(ds, ds') \wedge turn\_eq(ds', ds'') \rightarrow turn\_eq(ds, ds'') \\
\} \quad & \\
\equiv \quad & \\
\{ \quad & min \;\; turn\_eq : \\
& \widehat{turn}(ds', dr') \wedge ds = ds' \wedge dr = dr' \rightarrow turn\_eq(ds, dr) \\
& turn\_eq(ds, ds') \wedge turn\_eq(ds', ds'') \rightarrow turn\_eq(ds, ds'') \\
\} \quad & \\
\equiv \quad & \\
\{ \quad & min \;\; turn\_eq : \\
& \widehat{turn}(ds, dr) \rightarrow turn\_eq(ds, dr) \\
& turn\_eq(ds, ds') \wedge turn\_eq(ds', ds'') \rightarrow turn\_eq(ds, ds'') \\
\} \quad & \\
\equiv \quad & \\
& \widehat{turn} = turn\_eq
\end{aligned}$$

where the last equality follows from the fact that $\widehat{turn}$ is transitive, and $turn\_eq$ is the minimum transitive predicate containing $\widehat{turn}$.

From the definition of $travel\_eq$ (block 32), and a similar argument to the one above, we conclude that $travel\_eq = \widehat{travel}$. Consequently, we can dispense with the use of the predicates $turn\_eq$ and $travel\_eq$ in our formalization.

Therefore, if views uniquely identify distinctive states, **AT_block** (Page 9) can be rewritten as follows:

$$
\begin{array}{l}
AT\_block = \\
\quad \{ : \\
\qquad at(ds,p) \rightarrow tplace(p), \\
\qquad \exists! p\, at(ds,p), \\
\qquad \langle ds, turn, ds' \rangle \wedge at(ds,p) \rightarrow at(ds',p), \\
\qquad at(ds,p) \wedge at(ds',p) \rightarrow \widehat{turn}(ds,ds'), \\
\\
\qquad along(ds,pa,dir) \rightarrow tpath(pa), \\
\qquad at(ds,p) \wedge at(ds',q) \wedge \overrightarrow{travel}(ds,ds') \rightarrow \\
\qquad\quad \exists pa,\, dir \left\{ PO(pa,dir,p,q) \wedge along(ds,pa,dir) \wedge along(ds',pa,dir) \right\}, \\
\qquad along(ds,pa,dir) \wedge along(ds,pa1,dir1) \rightarrow pa = pa1, \\
\qquad at(ds,p) \wedge at(ds',p) \wedge along(ds,pa,dir) \wedge along(ds',pa,dir) \rightarrow ds = ds', \\
\\
\qquad along(ds,pa,dir) \wedge along(ds',pa1,dir1) \big\} \rightarrow pa \neq pa1, \\
\qquad \{ \langle ds, turn\_desc, ds' \rangle \wedge turn\_desc \neq turnAround \wedge \\
\qquad\quad along(ds,pa,dir) \wedge along(ds',pa1,dir1) \} \rightarrow pa \neq pa1, \\
\qquad \langle ds, turnAround, ds' \rangle \rightarrow along(ds,pa,dir) \equiv along(ds',pa,-dir), \\
\\
\qquad PO(pa,pos,p,q) \equiv PO(pa,neg,q,p), \\
\qquad \neg PO(pa,dir,p,p), \\
\qquad PO(pa,dir,p,q) \wedge PO(pa,dir,q,r) \rightarrow PO(pa,dir,p,r), \\
\qquad PO(pa,dir,p,q) \rightarrow OnPath(pa,p) \\
\qquad OnPath(pa,p) \wedge OnPath(pa,q) \wedge tpath(pa) \\
\qquad\qquad \rightarrow \exists ds, ds' \{ at(ds,p) \wedge at(ds',q) \wedge \widehat{travel}(ds,ds') \}, \\
\\
\qquad \textbf{circ}\ \ tpath \succ along \succ PO \succ OnPath \succ tplace\ \ \textbf{var}\ \ at \\
\quad \}
\end{array}
$$

F Logic Program Correctness

Given a set of experiences E, the models of the theory $CT(E)$ (Page 6) indicate under what circumstances it is possible to consider two distinctive states as referring to the same environment state. In order to calculate these models, next we define a logic program such that its answer sets are in a one to one correspondence with the models of $CT(E)$.

Recall that the theory $CT(E)$ is defined as follows:

$$
\begin{array}{ll}
CT(E) \; = & \\
\quad E,\ HS(E),\ DT & (45)\\
\quad \langle ds, a, ds' \rangle \wedge \langle ds, a, ds'' \rangle \rightarrow ds' = ds'' & (46)\\
\quad CEQ\_block \; = & \\
\quad \{\ max \;\; ceq : & \\
\quad\ \ ceq(ds, ds), & (47)\\
\quad\ \ ceq(ds, ds') \rightarrow ceq(ds', ds), & (48)\\
\quad\ \ ceq(ds, ds') \wedge ceq(ds', ds'') \rightarrow ceq(ds, ds''), & (49)\\
 & \\
\quad\ \ ceq(ds, ds') \rightarrow View(ds, v) \equiv View(ds', v), & (50)\\
\quad\ \ ceq(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle \rightarrow ceq(ds_1', ds_2') & (51)\\
\quad \} &
\end{array}
$$

The logic program Π we will consider is defined as follows:

$$
\begin{array}{rcll}
p(X,Y,X,Y) & \leftarrow & . & (52)\\
p(X,Y,X2,Y1) & \leftarrow & p(X,Y,X1,Y1), ceq(X1,X2). & (53)\\
p(X,Y,X1,Y2) & \leftarrow & p(X,Y,X1,Y1), ceq(Y1,Y2). & (54)\\
p(X,Y,X2,Y2) & \leftarrow & p(X,Y,X1,Y1), cs(X1,A,X2), cs(Y1,A,Y2). & (55)\\
p(X,Y,Y1,X1) & \leftarrow & p(X,Y,X1,Y1). & (56)\\
p(X,Y,X1,Y2) & \leftarrow & p(X,Y,X1,Y1), p(X,Y,Y1,Y2). & (57)\\
dist(X,Y) & \leftarrow & p(X,Y,X1,Y1), view(X1,V), not\ view(Y1,V). & (58)\\
dist(X,Y) & \leftarrow & p(X,Y,X1,Y1), not\ view(X1,V), view(Y1,V). & (59)\\
\multicolumn{3}{l}{ceq(X,Y); \neg ceq(X,Y) \leftarrow .} & (60)\\
 & \leftarrow & not\ ceq(X,X). & (61)\\
 & \leftarrow & ceq(X,Y), not\ ceq(Y,X). & (62)\\
 & \leftarrow & ceq(X,Y), ceq(Y,Z), not\ ceq(X,Z). & (63)\\
 & & & \\
 & \leftarrow & ceq(X,Y), view(X,V), not\ view(Y,V). & (64)\\
 & \leftarrow & ceq(X,Y), not\ view(X,V), view(Y,V). & (65)\\
 & \leftarrow & not\ ceq(X1,Y1), ceq(X,Y), cs(X,A,X1), cs(Y,A,Y1). & (66)\\
 & & & \\
 & \leftarrow & not\ ceq(X,Y), not\ dist(X,Y). & (67)
\end{array}
$$

where the variables Xs and Ys variable range over distinctive states and the variable V ranges over views. Rule 60 states that an answer set of the program should be *complete* with respect to ceq. Rules 61-63 require ceq to be an equivalence class. Rules 64 and 65 are the counterpart of axiom 50. Rule 66 is the counterpart of axiom 51. In order to define the maximality condition of ceq, the auxiliar predicate $p(X, Y, X1, Y1)$ is introduced. This predicate reads as *"If X and Y were the same, then $X1$ and $Y1$ would be the same"*. The predicate $dist(X, Y)$ defines when distinctive states X and Y are distinguishable. Constraint 67 establishes the maximality condition on ceq: $ceq(X, Y)$ should be the case unless X and Y are distinguishable.

Notation. Given a set of experiences E, $\Pi(E)$ denotes the grounded program consisting of the rules $\{cs(ds, a, ds') \leftarrow\ .\ :\ E \models \langle ds, a, ds'\rangle\}$, $\{view(ds, v) \leftarrow\ .\ :\ E \models View(ds, v)\}$, and replacing in Π the occurrences of the variables X, $X1$, $X2$, Y, $Y1$, $Y2$ and V by distinctive states and view symbols in E, respectively.

Given a model M for the axioms 45-51, $\Pi(E)(ceq^M)$ denotes the program consisting of $\Pi(E)$ and the rules $\{ceq(a, b) \leftarrow\ .\ :\ M \models ceq(a, b)\}$, $\{\neg ceq(a, b) \leftarrow\ .\ :\ M \not\models ceq(a, b)\}$.

Similarly, $\Pi(E)_1(ceq^M)$ denotes the program resulting by removing from $\Pi(E)(ce^M)$ the rules associated with grounding rule 67. By $AS(\Pi(E)_1(ceq^M))$ we denote the answer set of $\Pi(E)_1(ceq^M)$ (see lemma 2, page 32).

We say that ceq^M is *maximal* in M whenever M is a model for $CT(E)$.[26]

$\{$*end of notation*$\}$

Using the notation above we can state our theorem as follows:

Theorem 7 *ceq^M is* maximal *in M if and only if $AS(\Pi(E)_1(ceq^M))$ is an answer set for $\Pi(E)(ceq^M)$.*

Notice that theorem 7 establishes a one to one correspondence between the answer sets of $\Pi(E)$ and the models of $CT(E)$. Given a model M for $CT(E)$, ceq^M is *maximal* in M, and so $AS(\Pi(E)_1(ceq^M))$ is an answer set for $\Pi(E)(ceq^M)$, thus, an answer set for $\Pi(E)$.[27]Conversely, given any answer set X for $\Pi(E)$, the model M defined such that $M \models E$ and $ceq^M = \{(ds, ds') : (ds, ds') \in X\}$, is such that $AS(\Pi(E)_1(ceq^M)) = X$.[28] Consequently, ceq^M is *maximal* in M, that is, M is a model for $CT(E)$.

Proof of theorem 7.

(a) Suppose ceq^M is maximal in M and $AS(\Pi(E)_1(ceq^E))$ is **not** an answer set for $\Pi(E)(ceq^M)$. Since $AS(\Pi(E)_1(ceq^M))$ is an answer set for $\Pi(E)_1(ceq^M)$, then

[26]Recall that ceq^M denotes the interpretation of ceq in the structure M.

[27]Suppose, X is an answer set for $\Pi(E)(ceq^M)$ and consider $Y \subseteq X$ closed under $\Pi(E)$. Then, in virtue of 60, Y is closed under ceq^M and so $X = Y$.

[28]Given any program Π and X an answer set for Π, X is the unique answer set for $\Pi \cup X$.

$AS(\Pi(E)_1(ceq^M))$ does not satisfy constraint 67.[29] Consequently, there exist distinctive states X and Y such that:

1. $ceq(X, Y) \notin AS(\Pi(E)_1(ceq^M))$, and
2. $dist(X, Y) \notin AS(\Pi(E)_1(ceq^M))$.

Define $ceq1^M$ in M such that $M \models ceq1(a, b)$ whenever $M \models ceq(a, b)$ or $p(X, Y, a, b) \in AS(\Pi(E)_1(ceq^M))$. Symbolically,

$$ceq1^M(a, b) \equiv ceq^M(a, b) \vee p(X, Y, a, b) \ .$$

By definition, $ceq^M \subset ceq1^M$.[30] We are to prove that $ceq1^M$ satisfies axioms 47-51, which will contradict the fact that ceq^M is maximal in M:

- By lemma 3, $ceq1^M$ is an equivalence relation.
- Suppose that $ceq1^M(ds, ds')$ is the case. If $ceq^M(ds, ds')$ is the case, then $M \models View(ds, v) \equiv View(ds', v)$. If $p(X, Y, ds, ds')$ is the case, then $View(ds, v) \equiv View(ds', v)$ is the case, since otherwise $dist(X, Y)$ will belong to $AS(\Pi(E)_1(ceq))$.[31]
- Suppose that $M \models ceq1(ds_1, ds_2) \wedge \langle ds_1, a, ds_1' \rangle \wedge \langle ds_2, a, ds_2' \rangle$. If $M \models ceq(ds_1, ds_2)$ is the case, then $M \models ceq(ds_1', ds_2')$ is the case. If $p(X, Y, ds_1, ds_2)$ is the case, by rule 55, $p(X, Y, ds_1', ds_2') \in AS(\Pi(E)_1(ceq^M))$ and so $M \models ceq1(ds_1', ds_2')$.

□

(b) Suppose that $AS(\Pi(E)_1(ceq))$ is an answer set for $\Pi(E)(ceq^M)$ and ceq^M is *not* maximal in M. Then, there exists $ceq1^M$, $ceq^M \subset ceq1^M$, $ceq1^M$ maximal in M. Moreover, by the if part of this theorem, (a) above, $AS(\Pi(E)_1(ceq1^M))$ is an answer set for $\Pi(E)(ceq1^M)$. Let X and Y be such that:

1. $M \models ceq1(X, Y)$,
2. $M \not\models ceq(X, Y)$, and so (Lemma 2) $ceq(X, Y) \notin AS(\Pi(E)_1(ceq^M))$.

Since $AS(\Pi(E)_1(ceq^M))$ satisfies constraint 67, then $dist(X, Y) \in AS(\Pi(E)_1(ceq^M))$, and consequently (rules 58-59) there exist $X1$,$Y1$ and V, such that

1. $p(X, Y, X1, Y1) \in AS(\Pi(E)_1(ceq^M))$,
2. $M \models View(X1, V) \not\equiv View(Y1, V)$.

[29]The answer sets of $\Pi(E)(ceq^M)$ are those answer sets of $\Pi(E)_1(ceq^M)$ that satisfy constraint 67.

[30]Since $p(X, Y, X, Y) \in AS(\Pi(E)_1(ceq^M))$

[31]This is the case according to rules 58 and 59, and the fact that $AS(\Pi(E)_1(ceq))$ is the answer set for $\Pi(E)_1(ceq)$.

Since $ceq^M \subset ceq1^M$, then $p(X,Y,X1,Y1) \in AS(\Pi(E)_1(ceq1^M))$ (Lemma 4). Since $ceq(X,Y) \in AS(\Pi(E)_1(ceq1^M))$, then $ceq(X1,Y1) \in AS(\Pi(E)_1(ceq1^M))$ (Lemma 5) which is a contradiction since $AS(\Pi(E)_1(ceq1^M))$ satisfies constraints 64-65.

$\square$

Lemma 2 *Let M be a model for axioms 45-51. Let ceq^M be the interpretation of ceq in M.*[32] *Then $\Pi(E)_1(ceq^M)$ has a unique answer set which we denote by $AS(\Pi(E)_1(ceq^M))$. Moreover, for any two distinctive states ds and ds', $M \models ceq(ds,ds')$ if and only if $ceq(ds,ds') \in AS(\Pi(E)_1(ceq^M))$.*

Proof. Let $\Pi(E)_2(ceq^M)$ denote the program resulting of removing from $\Pi(E)_1(ceq^M)$ those constraints resulting from grounding rules 61-66. The answer sets of $\Pi(E)_1(ceq^M)$ are those answer sets of $\Pi(E)_2(ceq^M)$ satisfying constraints 61-66. We are to prove that $\Pi(E)_2(ceq^M)$ has a unique answer set satisfying constraints 52-59.

Let $Facts$ denote the union of the sets $\{cs(ds,a,ds') \leftarrow . : E \models \langle ds,a,ds'\rangle\}$, $\{view(ds,v) \leftarrow . : E \models View(ds,v)\}$, $\{ceq(a,b) \leftarrow . : M \models ceq(a,b)\}$, and $\{\neg ceq(a,b) \leftarrow . : M \not\models ceq(a,b)\}$. Any answer set of $\Pi(E)_2(ceq^M)$ contains $Facts$. Let X and Y denote two possible answer sets for $\Pi(E)_2(ceq^M)$. Then the reduct of X and Y are the same, $\Pi(E)_2(ceq^M)^X = \Pi(E)_2(ceq^M)^Y$, since both X and Y agree on the literals of the form $view(ds,v)$.[33] Consequently, $\Pi(E)_2(ceq^M)$ has at most one answer set. In fact, $Cn(\Pi(E)_2(ceq^M)^{Facts})$ is such answer set.[34] In particular, $ceq(ds,ds') \in Cn(\Pi(E)_2(ceq^M)^{Facts})$ iff $ceq(ds,ds') \in Facts$ iff $M \models ceq(ds,ds')$.

Finally, since ceq^M satisfies axioms 47-51 then $Cn(\Pi(E)_2(ceq^M)^{Facts})$ satisfies constraints 61-66, thus, it is an answer set for $\Pi(E)_1(ceq^M)$. $\square$

Lemma 3 *Let M be a model for axioms 45-51. Let ceq^M be the interpretation of ceq in M, and let $AS(\Pi(E)_1(ceq^M))$ be the answer set for $\Pi(E)_1(ceq^M)$. Let X and Y be two arbitrary distinctive state symbols. Let $ceq1^M$ in M be such that $M \models ceq1(a,b)$ whenever $M \models ceq(a,b)$ or $p(X,Y,a,b) \in AS(\Pi(E)_1(ceq^M))$. Symbolically,*

$$ceq1^M(a,b) \equiv ceq^M(a,b) \vee p(X,Y,a,b) \ .$$

Then, $ceq1^M$ is an equivalence relation.

Proof.

- $ceq1$ is reflexive. Indeed,

$$ceq1^M(ds,ds) \equiv ceq^M(ds,ds) \vee p(X,Y,ds,ds) \overset{47}{\equiv} ceq^M(ds,ds) \ .$$

[32]The interpretation of ceq in M is not necessarily maximal.

[33]Notice that $view(ds,ds') \in X$ iff $view(ds,ds') \in Facts$.

[34]$\mathbf{Cn(\Pi)}$ denotes the set of consequences of a logic program without negation as failure.

- $ceq1$ is symmetric. Indeed,

$$\begin{array}{rcl} ceq1^M(ds, ds') & \equiv & ceq^M(ds, ds') \vee p(X, Y, ds, ds') \\ & \overset{48}{\equiv} & ceq^M(ds', ds) \vee p(X, Y, ds, ds') \\ & \overset{56}{\equiv} & ceq^M(ds', ds) \vee p(X, Y, ds', ds) \\ & \equiv & ceq1^M(ds1, ds) \end{array}$$

- $ceq1$ is transitive. Indeed,

$$\begin{array}{ll} \multicolumn{2}{l}{ceq1^M(ds, ds') \wedge ceq1^M(ds', ds'')} \\ \equiv & \{ceq^M(ds, ds') \vee p(X, Y, ds, ds')\} \wedge \{ceq^M(ds', ds'') \vee p(X, Y, ds', ds'')\} \\ \equiv & \{ceq^M(ds, ds') \wedge ceq^M(ds', ds'')\} \vee \{ceq^M(ds, ds') \wedge p(X, Y, ds', ds'')\} \vee \\ & \{p(X, Y, ds, ds') \wedge ceq^M(ds', ds'')\} \vee \{p(X, Y, ds, ds') \wedge p(X, Y, ds', ds'')\} \\ \overset{49,57}{\rightarrow} & ceq^M(ds, ds'') \vee \{ceq^M(ds, ds') \wedge p(X, Y, ds', ds'')\} \vee \\ & \{p(X, Y, ds, ds') \wedge ceq^M(ds', ds'')\} \vee p(X, Y, ds, ds'') \\ \overset{53,54}{\rightarrow} & ceq^M(ds, ds'') \vee p(X, Y, ds, ds'') \vee p(X, Y, ds, ds'') \vee p(X, Y, ds, ds'') \\ \equiv & ceq^M(ds, ds'') \vee p(X, Y, ds, ds'') \\ \equiv & ceq1^M(ds, ds'') \end{array}$$

□

Lemma 4 *Let M be a model for axioms 45-51, and let ceq^M, $ceq1^M$, $ceq^M \subset ceq1^M$, be two relations such that both satisfy axioms 47-51. Then, $AS(\Pi(E)_1(ceq^M)) \subset AS(\Pi(E)_1(ceq1^M))$.*

Proof. Let $\Pi(E)_2(ceq^M)$ denote the program resulting of removing from $\Pi(E)_1(ceq^M)$ those constraints resulting from grounding rules 61-66. Let $Facts$ denote the union of the sets $\{cs(ds, a, ds') \leftarrow . : E \models \langle ds, a, ds' \rangle\}$, $\{view(ds, v) \leftarrow . : E \models view(ds, v)\}$, $\{ceq(a, b) \leftarrow . : M \models ceq(a, b)\}$, and $\{\neg ceq(a, b) \leftarrow . : M \not\models ceq(a, b)\}$. Define $Facts1$ in the same way as $Facts$ but using $ceq1$ instead of ceq. Since, $ceq^M \subset ceq1^M$, it is the case that $Facts \subset Fact1$ and consequently $\Pi(E)_2(ceq^M) \subset \Pi(E)_2(ceq1^M)$.

Since, $Facts$ and $Facts1$ agree on literals of the form $view(ds, v)$, it follows that $\Pi(E)_2(ceq^M)^{Facts} \subset \Pi(E)_2(ceq1^M)^{Facts1}$. It follows then that $Cn(\Pi(E)_2(ceq^M)^{Facts}) \subset Cn(\Pi(E)_2(ceq^M)^{Facts1}) \subset Cn(\Pi(E)_2(ceq1^M)^{Facts1})$. In lemma 2 we prove that $AS(\Pi(E)_1(ceq^M)) = Cn(\Pi(E)_2(ceq^M)^{Facts})$ and $AS(\Pi(E)_1(ceq1^M)) = Cn(\Pi(E)_2(ceq^M)^{Facts1})$. Therefore, $AS(\Pi(E)_1(ceq^M)) \subset AS(\Pi(E)_1(ceq1^M))$.
□

Lemma 5 *Let M be a model for axioms 45-51. If $ceq^M(ds, ds')$ and $p(ds, ds', ds1, ds1') \in AS(\Pi(E)_1(ceq^M))$, then $ceq^M(ds1, ds1')$.*

Proof. Consider the program $\hat{\Pi}(E)_1(ceq^M)$ resulting from $\Pi(E)_1(ceq^M)$, by removing those instances of rules 56 and 57 where $X = ds$ and $Y = ds'$. Let AS denote the answer set for $\hat{\Pi}(E)_1(ceq^M)$. We are to prove that $AS(\Pi(E)_1(ceq^M)) = AS$, which in virtue of property 68, proves our theorem. Notice that $AS(\Pi(E)_1(ceq^M))$ is closed under $\hat{\Pi}(E)_1(ceq^M))$, and consequently $AS \subseteq AS(\Pi(E)_1(ceq^M))$. We need to prove then that AS is closed under $\Pi(E)_1(ceq^M))$.

Since AS is the answer set for $\hat{\Pi}(E)_1(ceq^M)$, $p(ds, ds', X, Y) \in AS$ if and only if there exist distinctive states $X_0, \hat{X_0}, Y_0, \hat{X_0}, \ldots, X_n, \hat{X_n}, Y_n, \hat{X_n}$ and actions $A_0, \ldots, A_{n-1}$ such that

1. $(X_0, Y_0) = (ds, ds')$, $(\hat{X_n}, \hat{Y_n}) = (X, Y)$,

2. $ceq(X_i, \hat{X_i})$, $ceq(Y_i, \hat{Y_i})$, $0 \leq i \leq n$.

3. $cs(\hat{X_i}, A_i, X_{i+1})$, $cd(\hat{Y_i}, A_i, Y_{i+1})$, $0 \leq i < n$.

By induction on n we can prove then that

$$if\ p(ds, ds', X, Y) \in AS\ then\ ceq(X, Y) \in AS. \qquad (68)$$

For $n = 0$ we have that $(X_0, Y_0) = (ds, ds')$, $(\hat{X_0}, \hat{Y_0}) = (X, Y)$, $ceq(ds, X)$, and $ceq(ds', Y)$. Since ceq is an equivalence relation and $ceq(ds, ds')$, it follows that $ceq(X, Y)$. Suppose now that $(\hat{X_n}, \hat{Y_n}) = (X, Y)$ for some $n > 0$. By induction hypothesis, $ceq(\hat{X}_{n-1}, \hat{Y}_{n-1})$. Since ceq satisfies constraint 66, it follows that $ceq(X_n, Y_n) \in AS$. Since ceq is an equivalence relation, it follows then that $ceq(\hat{X_n}, \hat{Y_n})$.

Using 68 we prove that AS is closed under $\Pi(E)_1(ceq^M))$. Indeed,

$$\begin{array}{rcl} p(ds, ds', Y, X) & \overset{54}{\leftarrow} & p(ds, ds', Y, Y), ceq(Y, X) \\ & \overset{53,48}{\leftarrow} & p(ds, ds', X, Y), ceq(X, Y) \\ & \overset{68}{\leftarrow} & p(ds, ds', X, Y) \in AS \quad . \end{array}$$

$$\begin{array}{rcl} p(ds, ds', X, Z) & \overset{53}{\leftarrow} & p(ds, ds', Y, Z) \wedge ceq(Y, X) \\ & \overset{49}{\leftarrow} & p(ds, ds', X, Y) \wedge ceq(X, Y) \wedge \\ & & \qquad p(ds, ds', Y, Z) \wedge ceq(Y, Z) \\ & \overset{68}{\leftarrow} & p(ds, ds', X, Y) \in AS \wedge p(ds, ds', Y, Z) \in AS \quad . \end{array}$$

We have proved that AS is closed under $\Pi(E)_1(ceq^M))$. It follows that $AS = AS(\Pi(E)_1(ceq^M))$. $\Box$

References

[Basye *et al.*, 1995] K. Basye, T. Dean, and L. P. Kaelbling. Learning dynamics: System identification for perceptually challenged agents. *Artificial Intelligence*, 72(1):139–171, 1995.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Kortenkamp *et al.*, 1995] D. Kortenkamp, E. Chown, and S. Kaplan. Prototypes, locations, and associative networks (PLAN): towards a unified theory of cognitive mapping. *Cognitive Science*, 19:1–51, 1995.

[Kuipers and Byun, 1988] B. Kuipers and Y. T. Byun. A robust qualitative method for spatial learning in unknown environments. In Morgan Kaufmann, editor, *AAAI-88*, 1988.

[Kuipers and Byun, 1991] B. J. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[Kuipers, 1977] B. Kuipers. *Representing Knowledge of Large-Scale Space*. PhD thesis, Artificial Intelligence Laboratory, MIT, 1977.

[Kuipers, 1978] B. J. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[Kuipers, 2000] B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[Leiser and Zilbershatz, 1989] D. Leiser and A. Zilbershatz. THE TRAVELLER: a computational model of spatial network learning. *Environment and Behavior*, 21(4):435–463, 1989.

[Lifschitz, 1994] V. Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1994.

[Lifschitz, 1995] V. Lifschitz. Nested abnormality theories. *Artificial Intelligence*, (74):351–365, 1995.

[Lifschitz, 1999] V. Lifschitz. Answer set planning. *International Conference on Logic Programming*, 9:23–37, 1999.

[McCarthy, 1980] John McCarthy. Circumscription - A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 3(26):88–116, 1986.

[McDermott and Davis, 1984] D. V. McDermott and E. Davis. Planning routes through uncertain territory. *Artificial Intelligence*, 22:107–156, 1984.

[Niemelä and Simons, 1997] I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *4th International Conference on Logic Programming and Nonmonotonic Reasoning*, number 1265 in LNCS, pages 420–429. Springer-Verlag, 1997.

[Remolina and Kuipers, 1998] E. Remolina and B. Kuipers. Towards a formalization of the Spatial Semantic Hierarchy. In *Fourth Symposium on Logical Formalizations of Commonsense Reasoning, London*, January 1998.

[Remolina and Kuipers, 2001] E. Remolina and B. Kuipers. A logical account of causal and topological maps. In *International Joint Conference in Artificial Intelligence (IJCAI-01)*, August 2001. To appear.

[Shanahan, 1996] M. Shanahan. Noise and the common sense informatic situation for a mobile robot. In *AAAI-96*, pages 1098–1103, 1996.

[Shanahan, 1997] M. Shanahan. Noise, non-determinism and spatial uncertainty. In *AAAI-97*, pages 153–158, 1997.

[Yeap, 1988] W. K. Yeap. Towards a computational theory of cognitive maps. *Artificial Intelligence*, 34:297–360, 1988.