A knowledge-level approach for effective acting, sensing, and planning

Ron Petrick

Ph.D Thesis Presentation

Department of Computer Science
University of Toronto

Supervisor: Hector Levesque

22 August 2006
Motivation

- Reasoning about sensing as a form of action requires the ability to reason effectively about knowledge
  - Often essential for planning and high-level agent control
- Problem has been extensively studied and solutions often share a common approach
  - Reasoning about knowledge reduces to reasoning about an accessibility relation over a set of possible worlds
- Possible world models provide expressive representations
- Computational drawback: $n$ atomic formulae requires potentially $2^n$ distinct worlds
- Planning: must reason about sets of worlds to construct plans
Overview: this thesis

- Formally model knowledge and action (with sensing) in the situation calculus
- Focus on representations that support efficient reasoning yet are expressive
- An emphasis towards planning problems and the construction of a planning system
 Overview: two main parts

1. Relate two formal approaches in the situation calculus:
   - **SL theory** [Scherl & Levesque 1993]: knowledge of general first-order formulae using possible worlds
     \[ F(x, s), f(x, s) = y, K(s', s) \]
   - **DP theory** [Demolombe & Pozos Parra 2000]: literal-based knowledge modelled using “knowledge fluents”
     \[ F(x, s), KF(x, s), K\neg F(x, s) \]

2. Develop a “knowledge-level” approach to planning
   - Build plans based on what an agent knows
   - Limit representation for tractable reasoning
   - PKS planner: plan effectively in a rich collection of domains
   - Convert formal theories to PKS domains
Contributions

- A formal correlation between two existing representations of knowledge and action from the literature
  - Correctness of the knowledge fluent approach in established in terms of the standard possible worlds-based approach
  - Situations are identified where DP can be used in place of SL
  - Knowledge change reduces to ordinary fluent change
- Demonstrate the utility of knowledge-level planning with PKS
  - Describe a conditional planner with many useful features, e.g., functions, run-time variables, numerical reasoning
  - Expressiveness of the representation language permits a variety of problems to be solved
  - Empirical results demonstrate efficiency
- Preliminary results are provided for converting world-level representations into a knowledge-level form usable by PKS
Part I: Formal theory - Cartesian situations

To relate \( SL \) and \( DP \) we restrict the expressiveness of \( SL \) to prohibit some types of knowledge we can’t represent in \( DP \).

Impose a **Cartesian** structure over accessible situations

\[
\text{Cart}(s) \overset{\text{def}}{=} \left[ \bigwedge_{i=1}^{n} (\forall s', s'', \overline{x}_i). K(s', s) \land K(s'', s) \land F_i(\overline{x}_i, s') \neq F_i(\overline{x}_i, s'') \supset \right.

\( (\exists s^*). K(s^*, s) \land F_i(\overline{x}_i, s^*) \equiv F_i(\overline{x}_i, s'') \land (\forall \overline{x}_j)(\overline{x}_j \neq \overline{x}_i \supset F_i(\overline{x}_j, s^*) \equiv F_i(\overline{x}_j, s')) \land \right.

\[
\bigwedge_{k=1}^{n} (\forall \overline{x}_k) F_k(\overline{x}_k, s^*) \equiv F_k(\overline{x}_k, s') \land \bigwedge_{k=1}^{m} (\forall \overline{x}_k)f_k(\overline{x}_k, s^*) = f_k(\overline{x}_k, s') \bigwedge_{k \neq i}^{m}
\]

... 

Cartesian situations ensure there are “enough” accessible situations with “minimal” differences among the fluent values.

These restrictions give rise to a set of properties for manipulating knowledge of primitive fluent literals:

\( F(x, s), \neg F(x, s), f(x, s) = y, f(x, s) \neq y \)
Basic properties of Cartesian situations

Cartesian properties let us deconstruct complex $\text{Knows}_{SL}$ expressions into simpler “component” parts

1. $\text{Knows}_{SL} \left( \bigvee_{i=1}^{n} \theta_i(\text{now}), s \right) \equiv \bigvee_{i=1}^{n} \text{Knows}_{SL} (\theta_i(\text{now}), s),$

2. $\text{Knows}_{SL} (\theta_1(\text{now}) \supset \theta_2(\text{now}), s) \equiv \left( \text{Knows}_{SL} (\theta_1(\text{now}), s) \lor \text{Knows}_{SL} (\theta_2(\text{now}), s) \right),$

3. $\text{Knows}_{SL} (\theta_1(\text{now}) \equiv \theta_2(\text{now}), s) \equiv \left( \text{Knows}_{SL} (\theta_1(\text{now}), s) \land \text{Knows}_{SL} (\theta_2(\text{now}), s) \right) \lor \left( \text{Knows}_{SL} (\theta_1(\text{now}), s) \land \text{Knows}_{SL} (\theta_2(\text{now}), s) \right),$

4. $\text{Knows}_{SL} (f_1(\vec{x}_1, \text{now}) = f_2(\vec{x}_2, \text{now}), s) \equiv \left( \exists y \right). \text{Knows}_{SL} (f_1(\vec{x}_1, \text{now}) = y, s) \land \text{Knows}_{SL} (f_2(\vec{x}_2, \text{now}) = y, s),$

5. $\text{Knows}_{SL} (f_1(\vec{x}_1, \text{now}) \neq f_2(\vec{x}_2, \text{now}), s) \equiv \left( \forall y \right). \text{Knows}_{SL} (f_1(\vec{x}_1, \text{now}) \neq y, s) \lor \text{Knows}_{SL} (f_2(\vec{x}_2, \text{now}) \neq y, s).$

6. ...
Certain types of “propositional” $\text{Knows}_{SL}$ formulae can be completely broken down to knowledge of fluent literals.

Pairs of situations can be shown to be “mutually” Cartesian, given certain accessibility properties on $K$.

Cartesian structure is “brittle” and easily broken by actions.

For propositional $SL$ theories the problem of determining if a situation is Cartesian or not reduces to verifying

$$
\Sigma \models \text{Knows}_{SL}(\bigvee_{i=1}^{n} L_i(\text{now}), s) \supset \bigvee_{i=1}^{n} \text{Knows}_{SL}(L_i(\text{now}), s),
$$

for all collections of fluent literals (with no repeated fluents).
Knowledge equivalence in combined action theories

- An **$SL+DP$ combined action theory** contains axioms from both an $SL$ theory and a $DP$ theory.

- Describe a procedure for syntactically compiling $SL$ axioms into $DP$ axioms.

<table>
<thead>
<tr>
<th>$SL$</th>
<th>SSAs for $F$, $f$, $K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DP$</td>
<td>SSAs for $F$, $f$</td>
</tr>
<tr>
<td></td>
<td>SKSAs for $KF$, $K\neg F$, $Kf_{eq}$, $Kf_{neq}$</td>
</tr>
</tbody>
</table>

- Establish knowledge equivalence for "literal-based" knowledge:
  1. $\Sigma \models (\forall s)(\forall \vec{x}).Knows_{SL}(L(\vec{x}, \text{now}), s) \equiv KL(\vec{x}, s)$,
  2. $\Sigma \models (\forall s)(\forall \vec{x}, y).Knows_{SL}(f(\vec{x}, \text{now}) = y, s) \equiv Kf_{eq}(\vec{x}, y, s)$,
  3. $\Sigma \models (\forall s)(\forall \vec{x}, y).Knows_{SL}(f(\vec{x}, \text{now}) \neq y, s) \equiv Kf_{neq}(\vec{x}, y, s)$.

- Consider theories that preserve the Cartesian structure through action, i.e., $\Sigma \models (\forall s)\text{Cart}(s)$. 
Transformation (compilation) process

- Must compile the “knowledge effects” of all actions on $F(f)$ into the SKSAs for $KF$ and $K\neg F$ ($K_{eq}$ and $K_{neq}$)

- Sensing actions
  - Restrict the form of the $K$ SSA to simplify the transformation; allow guarded effects
  - Distribute effects from $K$ axiom to appropriate SKSAs
  - Syntactically manipulate structure of $K$ axiom

- Physical actions
  - Provide a general transformation of SSAs into corresponding knowledge fluent versions
  - Syntactically change $L$ to $KL$; $f(\vec{x}, s) = t$ to $K_{eq}(\vec{x}, t, s)$; $f(\vec{x}, s) \neq t$ to $K_{neq}(\vec{x}, t, s)$
Transformation of sensing actions

\[ K(s'', \text{do}(a, s)) \equiv (\exists s'). s'' = \text{do}(a, s') \land K(s', s) \land \varphi_1(a, s, s') \land \varphi_2(a, s, s') \land \varphi_3(a, s, s'), \]

Sense \( F(x) \) for a specific \( x \):

\[ \varphi_1(a, s, s') \overset{\text{def}}{=} (\forall x) \ (a = \text{sense}_1(x) \supset (F(x, s) \equiv F(x, s'))), \]

Sense \( F(x) \) for every value of \( x \):

\[ \varphi_2(a, s, s') \overset{\text{def}}{=} (\forall x) \ (a = \text{sense}_2 \supset (F(x, s) \equiv F(x, s'))), \]

Sense \( f(x) \) and conditionally sense \( F(x) \):

\[ \varphi_3(a, s, s') \overset{\text{def}}{=} (\forall x) \ (a = \text{sense}_3(x) \supset (f(x, s) = f(x, s')) \land \\
(\forall x) \ (a = \text{sense}_3(x) \supset (F(x, s) \equiv F(x, s'))). \]
Transformation of physical actions

- Form classes of $SL + DP$ theories by restricting SSAs
  1. Context-free theories

  \[\text{broken}(x, do(a, s)) \equiv (a = \text{drop}(x) \land \text{fragile}(x)) \lor \text{broken}(x, s) \land \neg (a = \text{repair}(x)).\]

  2. Fluent dependent theories

  \[\text{on}(do(a, s)) \equiv a = \text{flip} \land \neg \text{on}(s) \lor \text{on}(s) \land \neg (a = \text{flip} \land \text{on}(s)).\]

  3. Literal-based theories

  \[\text{indir}(f, d, do(a, s)) \equiv (\exists d')(a = \text{mv}(f, d', d) \land d \neq d' \land \text{indir}(f, d', s)) \lor \text{indir}(f, d, s) \land \neg (\exists d')(a = \text{mv}(f, d, d') \land d \neq d').\]

- Establish a literal-based knowledge equivalence; preserve the Cartesian property through action
Extending knowledge equivalence to general formulae

- Introduce notion of $\text{Knows}_{DP}$:
  - $\text{Knows}_{DP}(F(\vec{x}), s) \overset{\text{def}}{=} KF(\vec{x}, s),$
  - $\text{Knows}_{DP}(\neg F(\vec{x}), s) \overset{\text{def}}{=} K\neg F(\vec{x}, s),$
  - $\text{Knows}_{DP}(f(\vec{x}) = t, s) \overset{\text{def}}{=} Kf_{eq}(\vec{x}, t, s),$
  - $\text{Knows}_{DP}(f(\vec{x}) \neq t, s) \overset{\text{def}}{=} Kf_{neq}(\vec{x}, t, s),$
  - $\text{Knows}_{DP}(\phi \land \psi, s) \overset{\text{def}}{=} \text{Knows}_{DP}(\phi, s) \land \text{Knows}_{DP}(\psi, s),$
  - $\text{Knows}_{DP}((\forall \vec{x}).\phi, s) \overset{\text{def}}{=} (\forall \vec{x})\text{Knows}_{DP}(\phi, s),$
  - $\ldots$

- Extend knowledge equivalence to more general $\phi$:

  \[ \Sigma \models \text{Knows}_{SL}(\phi(\text{now}), s) \equiv \text{Knows}_{DP}(\phi(\text{now}), s) \]

- Requires $\text{Cart}(s)$ to “break apart” $SL$ knowledge
- Requires literal-based knowledge equivalence to equate components
Additional results

- **$SL + DP$ theories**
  - $K$ axiom preserves accessibility properties through action
  - Compare “size” of compiled $DP$ theory to original $SL$ theory
  - Non-equivalence (e.g., knowledge-reducing actions, functions)

- **$DP + FI$ theories [Funge 1998]**
  - Establish knowledge equivalence between (relational) knowledge fluents and interval-valued fluents ranging over $\mathbb{B}$

  $KF(s) \iff IF(s) = \langle 1, 1 \rangle$
  $K\neg F(s) \iff IF(s) = \langle 0, 0 \rangle$
  $\neg KF(s) \land \neg K\neg F(s) \iff IF(s) = \langle 0, 1 \rangle$

  - Syntactic transformation of general successor knowledge state axioms for $KF, K\neg F$ into successor state axiom for $IF$

- Compare with closely related work
  (Baral & Son, Funge, Liu & Levesque, Soutchanski)
Part II: Knowledge-level planning

- Motivations are similar to motivations for formal theory
  - Restrict the representation for tractable reasoning
  - Work directly with knowledge level formulae
  - Build plans based on what is known at plan time

- Allow sensing actions and conditional plans to manage incomplete information

- Goal: plan effectively without manipulating individual possible worlds
The agent’s knowledge is represented by a set of databases, each of which models a particular type of knowledge.

The database contents (DB) have a fixed translation to formulae in a modal logic of knowledge (KB).

Primitive query language: $K(\alpha), K(\neg\alpha), K_w(\alpha), K_v(t), \neg\ldots$

A sound but incomplete inference procedure (IA) checks the database contents to determine the truth of a query.

Representation supports STRIPS/ADL-style actions that can update any of the databases with add/del operations.

- Update knowledge state rather than world state.

Supports functions and run-time variables.
PKS: Planning with Knowledge and Sensing

- PKS uses a collection of databases to represent its knowledge:
  - $K_f$: knowledge of positive and negative facts (no CWA)
    
    $p(a) \neg q(b, c) \quad f(a) = c \quad g(b, c) \neq d$
  
  - $K_w$: plan-time knowledge of sensing effects
    
    $\phi(\vec{x}) \in K_w$: know $\phi(\vec{x})$ or know $\neg\phi(\vec{x})$ at execution
  
  - $K_v$: plan-time knowledge of function values
    
    $f(\vec{x}) \in K_v$: know $f(\vec{x})$’s value at execution
  
  - $K_x$: exclusive-or knowledge
    
    $(\ell_1|\ell_2|\ldots|\ell_n)$: exactly one of the $\ell_i$ must be true

- Can specify rules to model knowledge-level state invariants; databases are automatically kept mutually consistent.
Actions in PKS

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconditions</th>
<th>Effects</th>
</tr>
</thead>
</table>
| pour-on-lawn    | $K(\text{haveBottle})$ | $\neg K(\neg \text{poisonous}) \Rightarrow$  
|                 |                | del($K_f, \neg \text{lawn-dead}$)           
|                 |                | $K(\text{poisonous}) \Rightarrow$           
|                 |                | add($K_f, \text{lawn-dead}$)                |
| sense-lawn      |                | $\text{add}(K_w, \text{lawn-dead})$        |

- ** Preconditions:** a conjunction of primitive queries
- ** Effects:** conditional-effect rules of the form $C \Rightarrow E$
  - $C$ is a conjunction of primitive queries
  - $E$ are $\text{add}$ and $\text{del}$ operations to any of the databases
- **Easy to compute new knowledge states:**
  - Queries are evaluated with $\text{IA}$ against a $\text{DB}$
  - Actions update $\text{DB} \Rightarrow \text{update KB}$
PKS generates conditional plans by forward-chaining:

Branches are constructed from $K_w$ formulae that resolve to definite knowledge at run time.

Goals are guaranteed to be satisfied along every branch.

PKS plans satisfy Levesque’s criteria for plan correctness:
- Plan time: the planner must know it will have enough information at run time for the plan to achieve the goals.
- Run time: the planner must have sufficient knowledge at every step of the plan to execute it.
Extensions to basic PKS

- Restrictions on the databases mean that certain types of knowledge cannot be modelled (e.g., general disjunctions)
  - Certain types of planning problems cannot be represented

- Representation and planning capabilities of basic PKS are extended in four areas:
  1. Enhanced inference mechanism (postdiction),
  2. Complex queries and temporally-extended goals
  3. Numerical expressions
  4. Finite-range functions
1 When PKS generates a conditional plan...

...it may fail to make certain “intuitive” conclusions.

2 Form linearizations of the possible execution branches:
Augment the states by applying 4 new inference rules:

Conclusions don’t follow solely from action effects but also require reasoning about action non-effects

- Worst case: $O(nd^2)$ testings of the rules ($n$ leaves, depth $d$)
- A formal interpretation of postdiction is given in terms of the situation calculus
Goal queries permit a temporal component:
- $Q^N$: query final state (e.g., classical goals)
- $Q^0$: query initial state (e.g., restore goals)
- $Q^\ast$: query all states (e.g., “hands-off” goals)

Queries can be combined into (restricted) formulae involving conjunction, disjunction, negation, and quantification.

A subset of C language expressions are permitted in queries and DB updates:
- Standard arithmetic operations
- Logical connectives, temporary variables
- Control structures (e.g., if-else, iterative loops)

Proviso: must be able to evaluate to a number at plan time

PKS can construct a multi-way branch in a plan with $K_v$ knowledge of a finite-range function
A syntactic transformation lets us compile “simple” DP theories into PKS using a (restricted) normal form
- Extract effect portions from SKSAs
- Convert to PKS database queries and updates
- Convert from an “online” to an “offline” form

<table>
<thead>
<tr>
<th>DP effect axiom</th>
<th>PKS action effects: $A(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = A(x) \supset KF(x, do(a, s))$</td>
<td>$add(K_f, F(x))$</td>
</tr>
<tr>
<td>$a = A(x) \land KP(x, s) \supset K\neg G(do(a, s))$</td>
<td>$K(P(x)) \Rightarrow add(K_f, \neg G)$</td>
</tr>
<tr>
<td>$a = A(x) \land H(x) \supset KH(x, do(a, s))$</td>
<td>$add(K_w, H(x))$</td>
</tr>
<tr>
<td>$a = A(x) \land \neg H(x) \supset K\neg H(x, do(a, s))$</td>
<td></td>
</tr>
</tbody>
</table>

An important step towards automatically converting world-level actions to knowledge-level actions (usable by PKS) with DP as an “intermediate” representation.
PKS software

Command-line interface

Graphical user interface

Application program

API

libPKS

Core library

XML + PKSL

Domain descriptor files

User Interfaces

DP to PKS
Planning problems and empirical results

- Standard benchmarks
  - Bomb in the toilet (easy at the knowledge level)
  - Medicate ($K_x$ knowledge)
- Opening a safe (functions)
  - Multiple combinations
    ($K$ preconditions as search control; natural plans generated)
  - Single known combination
    ($K_v$ knowledge as a run-time variable; $n$-way branching)
- Poisonous liquid (postdiction, temporally-extended goals)
- Painted door (postdiction, temporally-extended goals)
- UNIX domain
  - Finding a file
  - Finding the largest copy of a file
  - Preserving directory permissions
Conclusions and contributions

- A formal correlation is established between two existing representations of knowledge and action from the literature
  - Establish correctness of the \( DP \) approach in terms of the standard possible worlds-based \( SL \) approach
  - Identify classes where the computationally more efficient approach can be used in place of the possible world one
  - Knowledge change reduces to ordinary fluent change
  - Extend primitive equivalence to more general formulae

- Formally characterize properties of Cartesian situations

- Illustrate utility of knowledge-level planning with PKS
  - Planner supports many useful features like functions, run-time variables, postdiction, numerical reasoning
  - Empirical results – “abstracted” knowledge-level reasoning can be very efficient and sometimes leads to “natural” plans
  - PKS solves an interesting range of problems

- Conversion from world level \( \Rightarrow \) knowledge level \( \Rightarrow \) PKS
Publications

R. Petrick and H. Levesque (KR-2002),
Knowledge equivalence in combined action theories.

R. Petrick and F. Bacchus (AIPS-2002),
A knowledge-based approach to planning with incomplete information and sensing.

R. Petrick and F. Bacchus (ICAPS-2004),
Extending the knowledge-based approach to planning with incomplete information and sensing.

R. Petrick and F. Bacchus (ICAPS-2003 Workshop),
Reasoning with conditional plans in the presence of incomplete knowledge.

R. Petrick and F. Bacchus (ICAPS-2004 System Demo),
PKS: Knowledge-based planning with incomplete information and sensing.