Event Calculus and Answer Set Programming

Joohyung Lee

Automated Reasoning Group
School of Computing, Informatics and Decision Systems Engineering
Arizona State University

KOCSEA 2009

(Based on IJCAI 2007, 2009 Papers)
Knowledge in Action
Knowledge in Action

How do we prove .... ?
Challenges

- How do we know that the box does not move while the monkey walks to it? (Frame Problem)
Challenges

- How do we know that the box does not move while the monkey walks to it? (Frame Problem)
- How do we know that the monkey can move the box? (Qualification Problem)
Challenges

▶ How do we know that the box does not move while the monkey walks to it? (Frame Problem)
▶ How do we know that the monkey can move the box? (Qualification Problem)
▶ How do we know that the bananas move along with the monkey? (Ramification Problem)
Nonmonotonic Reasoning

- Human level intelligence requires **defeasible reasoning**: conclusions are drawn tentatively using **default assumptions** and can be retracted in the light of further information.

- Difficult to handle in first-order logic.
  - Because first-order logic is **monotonic**: if $\Gamma \vdash A$, then $\Gamma \cup \Delta \vdash A$.
  - Need for **nonmonotonic reasoning**: $\Gamma \vdash A$, but possibly $\Gamma \cup \Delta \nvdash A$. 
## Action Formalisms: Modelling Dynamicity of Systems

<table>
<thead>
<tr>
<th>High level</th>
<th>Low level</th>
<th>Foundation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Situation Calculus</td>
<td>Circumscription</td>
<td>Classical Logic</td>
</tr>
<tr>
<td>Event Calculus</td>
<td>Completion</td>
<td></td>
</tr>
<tr>
<td>[Kowalski &amp; Sergot, 1986]</td>
<td>[Clark, 1978]</td>
<td></td>
</tr>
<tr>
<td>Temporal Action Logics</td>
<td>Stable Model Semantics</td>
<td></td>
</tr>
<tr>
<td>[Doherty, 1996]</td>
<td>[Gelfond &amp; Lifschitz, 1988]</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td>Nonmonotonic Causal Theories</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Lee, et al., 2004]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>Action Languages</td>
<td>Default Logic [Reiter, 1980]</td>
<td></td>
</tr>
<tr>
<td>(A) [Gelfond &amp; Lifschitz, 1993]</td>
<td>Stable Model Semantics</td>
<td></td>
</tr>
<tr>
<td>(C) [Giunchiglia &amp; Lifschitz, 1998]</td>
<td>Nonmonotonic Causal Theories</td>
<td></td>
</tr>
<tr>
<td>(C^+) [Lee, et al., 2004]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- McCarthy & Hayes, 1969
- Gelfond & Lifschitz, 1993
- Giunchiglia & Lifschitz, 1998
- Kowalski & Sergot, 1986
- Doherty, 1996
- Reiter, 1980
- Gelfond & Lifschitz, 1988
- Clark, 1978
- Lee, et al., 2004
Event Calculus

Original version by Kowalski and Sergot, 1986.

Applied to database updates, robotics, policy-based computing, web service composition, natural language understanding, video games, ...
Stable Model Semantics

- Mathematical basis of answer set programming, a new form of declarative logic programming oriented towards combinatorial search problems and knowledge-intensive applications.

- Applied to planning, diagnosis, decision support systems, model checking, production configuration, VLSI routing, the Semantic Web, computational linguistics, bioinformatics, ...
Nonmonotonic Logics and classical logic are closely related.

Synergies can be obtained.
Initiates(Walk(l), At(Monkey, l), t)

HoldsAt(At(Monkey, l₁), t)
→ Terminates(Walk(l₂), At(Monkey, l₁), t)

Happens(Walk(l), t)
→ ¬HoldsAt(At(Monkey, l), t) ∧ ¬HoldsAt(OnBox, t)

HoldsAt(HasBananas, t) ∧ HoldsAt(At(Monkey, l), t)
→ HoldsAt(At(Bananas, l), t)

....
Monkey and Bananas in Event Calculus

\[\text{Initiates}(Walk(l), \text{At}(\text{Monkey}, l), t)\]

\[\text{HoldsAt}(\text{At}(\text{Monkey}, l_1), t) \rightarrow \text{Terminates}(Walk(l_2), \text{At}(\text{Monkey}, l_1), t)\]

\[\text{Happens}(Walk(l), t) \rightarrow \neg\text{HoldsAt}(\text{At}(\text{Monkey}, l), t) \land \neg\text{HoldsAt}(\text{OnBox}, t)\]

\[\text{HoldsAt}(\text{HasBananas}, t) \land \text{HoldsAt}(\text{At}(\text{Monkey}, l), t) \rightarrow \text{HoldsAt}(\text{At}(\text{Bananas}, l), t)\]

\[\cdots\]

We need to add domain independent axioms in the Event Calculus.
Monkey and Bananas in Event Calculus

\[ \text{Initiates}(Walk(l), At(Monkey, l), t) \]

\[ \text{HoldsAt}(At(Monkey, l_1), t) \rightarrow \text{Terminates}(Walk(l_2), At(Monkey, l_1), t) \]

\[ \text{Happens}(Walk(l), t) \rightarrow \neg \text{HoldsAt}(At(Monkey, l), t) \land \neg \text{HoldsAt}(OnBox, t) \]

\[ \text{HoldsAt}(HasBananas, t) \land \text{HoldsAt}(At(Monkey, l), t) \rightarrow \text{HoldsAt}(At(Bananas, l), t) \]

....

We need to add domain independent axioms in the Event Calculus.

We need to minimize the extents of \textit{Initiates}, \textit{Terminates}, \textit{Happens}. 
Domain Independent Event Calculus Axioms

\[
\text{Happens}(e, t) \land \text{Initiates}(e, f, t) \rightarrow \text{HoldsAt}(f, t+1)
\]

\[
\text{HoldsAt}(f, t) \land \neg \text{ReleasedAt}(f, t+1) \land
\neg \exists e (\text{Happens}(e, t) \land \text{Terminates}(e, f, t)) \rightarrow \text{HoldsAt}(f, t+1)
\]

....
In order to minimize the extents of \textit{Initiates}, \textit{Terminates}, \textit{Releases}, \textit{Happens}.

$\text{CIRC}[F; p]$ is a second-order formula such that its models are the models of $F$ that are minimal on $p$.

$$\text{CIRC}[F; p] = F \wedge (2\text{nd-order formula that makes } p \text{ minimal})$$
An event calculus domain description is defined as

$$\text{CIRC}[\Sigma; \text{Initiates, Terminates, Releases}] \land \text{CIRC}[\Delta; \text{Happens}] \land F$$

where

- $\Sigma$ is a conjunction of “effect” axioms
  - $[\text{condition}] \rightarrow \text{Initiates}(e, f, t)$
  - $[\text{condition}] \rightarrow \text{Terminates}(e, f, t)$
  - $[\text{condition}] \rightarrow \text{Releases}(e, f, t)$

- $\Delta$ is a conjunction of “event occurrence” axioms
  - $\text{Happens}(e, t)$

- $F$ is a conjunction of first-order sentences describing UNA, observations and the event calculus axioms.
Event Calculus Reasoning Tools

- Prolog
- Abductive logic programming [Eshghi, 1988; Shanahan, 1989]
- DEC REASONER (SAT-based) by Mueller (IBM)
  http://decreasoner.sourceforge.net/csr/ecas/

- ECASP REASONER (ASP-based) by us
  http://reasoning.eas.asu.edu/ecasp
What is Answer Set Programming (ASP)?

A new form of declarative programming oriented towards combinatorial search problems and knowledge-intensive applications.

The idea of ASP is to represent a given search problem as the problem of finding an answer set for some logic program, and then find a solution using an answer set solver.
num(1..8).
#domain num(I;I1;J;J1).

1 {q(I,J): num(J)} 1.
:- q(I,J), q(I1,J), I<I1.
:- q(I,J), q(I1,J1), I<I1, I1-I==abs(J1-J).

Given the input, an ASP solver can return 92 answer sets, which correspond 1-1 with 92 solutions.
Finding One Solution for the 8-Queens Problem

With command line

smodels version 2.26. Reading...done
Answer: 1
Stable Model: q(1,4) q(2,6) q(3,1) q(4,5) q(5,2)
q(6,8) q(7,3) q(8,7)
True
Duration 0.020
Number of choice points: 3
Number of wrong choices: 0
Number of atoms: 89
Number of rules: 552
Number of picked atoms: 240
Number of forced atoms: 5
Number of truth assignments: 2535
Size of searchspace (removed): 64 (0)
Finding All Solutions for the 8-Queens Problem

With the same program, but with the following command line

% lparse 8queen | smodels 0

smodels computes and shows all 92 valid queen arrangements. For instance, the last one is

Answer: 92
Stable Model: q(1,7) q(2,2) q(3,4) q(4,1) q(5,8)
q(6,5) q(7,3) q(8,6)
Brief History of Answer Set Programming

1988: Definition of answer sets for Prolog-like programs.
1992: Extending the definition to more general programs.
1999: ASP identified as a new programming paradigm.

WASP (Working Group on Answer Set Programming) : 17 European universities in 8 countries. Funded by EU.

Conferences/workshops : LPNMR, ASP, ASPOCP, LaSh, NMR. Biennial ASP solver competition from 2007.

The original paper [Gelfond and Lifschitz, ICLP 1988] is the 29th most cited Computer Science articles as of February 26, 2008 according to citeseer.
Relationship Between Two Traditions

Event Calculus
Situation Calculus (some versions)
Temporal Action Logics

<table>
<thead>
<tr>
<th>based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circumscription [1980, 1986]</td>
</tr>
</tbody>
</table>

[Lin, 1991]
[Ferraris et al., 2007]

Action Languages

<table>
<thead>
<tr>
<th>based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable Model Semantics [1988, 2007]</td>
</tr>
</tbody>
</table>

[Lee and Lin, 2006] (propositional case)
This paper (first-order case)
(Informally)

**Theorem 1**  Circumscription can be embedded into the stable model semantics.

**Theorem 2**  Event calculus can be reformulated in terms of the stable model semantics, and can be computed by ASP solvers.
Example: Turning Event Calculus Description to ASP

\[(\text{HoldsAt}(f, t) \land \neg\text{ReleasedAt}(f, t+1) \land \\
\neg\exists e (\text{Happens}(e, t) \land \text{Terminates}(e, f, t)) \rightarrow \text{HoldsAt}(f, t+1)\].

is turned into the conjunction of

\[(\text{HoldsAt}(f, t) \land \neg\text{ReleasedAt}(f, t+1) \land \\
\neg q(f, t)) \rightarrow \text{HoldsAt}(f, t+1)\\
\text{Happens}(e, t) \land \text{Terminates}(e, f, t) \rightarrow q(f, t)\]

and then turned into rules

\[\text{HoldsAt}(f, t+1) \leftarrow \text{HoldsAt}(f, t), \text{not ReleasedAt}(f, t+1), \\
\text{not } q(f, t)\]

\[q(f, t) \leftarrow \text{Happens}(e, t), \text{Terminates}(e, f, t)\]
ECASP vs. DEC reasoner

http://reasoning.eas.asu.edu/ecasp

http://decreasoner.sourceforge.net/csr/ecas/
ASP-based vs. SAT-based Approach

- **DEC** reasoner is based on the reduction of circumscription to completion. Able to solve 11 out of 14 benchmark problems.
- **ECASP** can handle the *full* version of the event calculus (modulo grounding). Able to solve all 14 problems.
- **ECASP** computes faster.
## Experiments (I)

<table>
<thead>
<tr>
<th>Problem</th>
<th>DEC reasoner</th>
<th>ECASP w/ LPARSE + CMODELS</th>
<th>ECASP w/ GRINGO + CLASP</th>
<th>ECASP w/ CLINGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusRide (15)</td>
<td>—</td>
<td>0.48 (0.42+0.06) A:156/R:7899/C:188</td>
<td>0.04 (0.03+0.01) A:733/R:3428</td>
<td>—</td>
</tr>
<tr>
<td>Commuter (15)</td>
<td>—</td>
<td>498.11 (447.50+50.61) A:4913/R:7383943/C:4952</td>
<td>44.42 (37.86+6.56) A:24698/R:5381620</td>
<td>28.79</td>
</tr>
<tr>
<td>Kitchen Sink (25)</td>
<td>71.10 (70.70+0.40) A:1014/C:12109</td>
<td>43.17 (37.17+6.00) A:123452/R:482018/C:0</td>
<td>2.47 (1.72+0.75) A:114968/R:179195</td>
<td>2.03</td>
</tr>
<tr>
<td>Thielser Circuit (20)</td>
<td>13.9 (13.6+0.3) A:5138/C:16122</td>
<td>0.42 (0.38+0.04) A:3160/R:9131/C:0</td>
<td>0.07 (0.05+0.02) A:1686/R:6510</td>
<td>0.05</td>
</tr>
<tr>
<td>Walking Turkey (15)</td>
<td>—</td>
<td>0.05 (0.04+0.01) A:556/R:701/C:0</td>
<td>0.04 (0.01+0.03) A:364/R:503</td>
<td>0.01</td>
</tr>
</tbody>
</table>

A: number of atoms, C: number of clauses, R: number of ground rules

**DEC reasoner and CMODELS used the same SAT solver RELSAT.**
## Experiments (II)

<table>
<thead>
<tr>
<th>Problem (max. time)</th>
<th>DEC reasoner</th>
<th>ECASP w/ LPARSE + CMODELS</th>
<th>ECASP w/ GRINGO + CLASP</th>
<th>ECASP w/ CLINGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falling w/ AntiTraj (15)</td>
<td>270.2 (269.3+0.9)</td>
<td>0.74 (0.66+0.08)</td>
<td>0.10 (0.08+0.02)</td>
<td>0.08</td>
</tr>
<tr>
<td>Falling w/ Events (25)</td>
<td>107.70 (107.50+0.20)</td>
<td>34.77 (30.99+3.78)</td>
<td>2.90 (2.01+0.89)</td>
<td>2.32</td>
</tr>
<tr>
<td>HotAir Baloon (15)</td>
<td>61.10 (61.10+0.00)</td>
<td>0.19 (0.16+0.03)</td>
<td>0.04 (0.03+0.01)</td>
<td>0.03</td>
</tr>
<tr>
<td>Telephone1 (40)</td>
<td>18.00 (17.50+0.50)</td>
<td>1.70 (1.51+0.19)</td>
<td>0.31 (0.26+0.05)</td>
<td>0.25</td>
</tr>
</tbody>
</table>

A: number of atoms, C: number of clauses, R: number of ground rules
Nonmonotonic logics and classical logic are closely related to each other. Synergies can be obtained by combining them. ASP solvers can be used as a general reasoning engine for circumscription based approaches, such as circumscriptive event calculus. This approach can handle the full version of the event calculus, modulo grounding.
Acknowledgements

NSF-IIS 0839821, 0916116
IARPA SCIL Grant