

# Event Calculus and Answer Set Programming

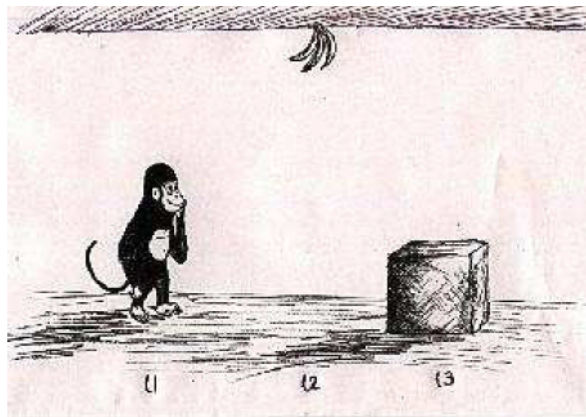
Joohyung Lee

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

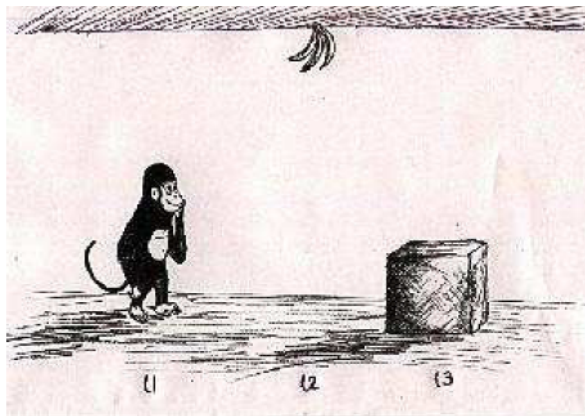
KOCSEA 2009

(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  $\Gamma \cup \Delta \not\vdash A$ .

# Action Formalisms: Modelling Dynamicity of Systems

## High level

Situation Calculus

[McCarthy & Hayes, 1969]

Event Calculus

[Kowalski & Sergot, 1986]

Temporal Action Logics

[Doherty, 1996]

...

Action Languages

$\mathcal{A}$  [Gelfond & Lifschitz, 1993]

$\mathcal{C}$  [Giunchiglia & Lifschitz, 1998]

$\mathcal{C}+$  [Lee, *et al.*, 2004]

...

## Low level

Circumscription

[McCarthy, 1980;1986]

Completion

[Clark, 1978]

Default Logic [Reiter, 1980]

Stable Model Semantics

[Gelfond & Lifschitz, 1988]

Nonmonotonic Causal Theories

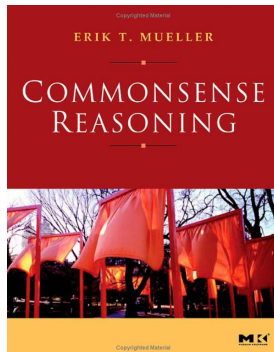
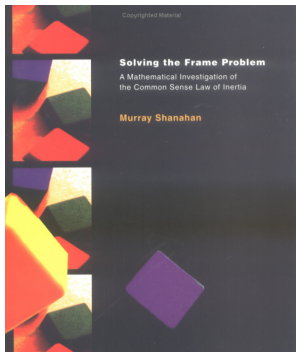
[Lee, *et al.*, 2004]

## Foundation

Classical Logic

Nonmonotonic Logics

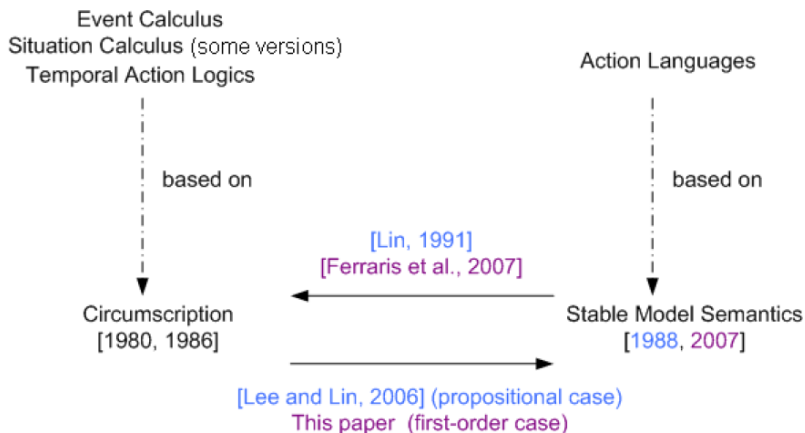
Original version by Kowalski and Sergot, 1986.



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

- ▶ 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, ...

# This Talk is about....



- ▶ Nonmonotonic Logics and classical logic are closely related.
- ▶ Synergies can be obtained.

# Monkey and Bananas in Event Calculus

$Initiates(Walk(l), At(Monkey, l), t)$

$HoldsAt(At(Monkey, l_1), t)$

$\rightarrow Terminates(Walk(l_2), At(Monkey, l_1), t)$

$Happens(Walk(l), t)$

$\rightarrow \neg HoldsAt(At(Monkey, l), t) \wedge \neg HoldsAt(OnBox, t)$

$HoldsAt(HasBananas, t) \wedge HoldsAt(At(Monkey, l), t)$

$\rightarrow HoldsAt(At(Bananas, l), t)$

....

# Monkey and Bananas in Event Calculus

$Initiates(Walk(l), At(Monkey, l), t)$

$HoldsAt(At(Monkey, l_1), t)$

$\rightarrow Terminates(Walk(l_2), At(Monkey, l_1), t)$

$Happens(Walk(l), t)$

$\rightarrow \neg HoldsAt(At(Monkey, l), t) \wedge \neg HoldsAt(OnBox, t)$

$HoldsAt(HasBananas, t) \wedge HoldsAt(At(Monkey, l), t)$

$\rightarrow HoldsAt(At(Bananas, l), t)$

....

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

# Monkey and Bananas in Event Calculus

$Initiates(Walk(l), At(Monkey, l), t)$

$HoldsAt(At(Monkey, l_1), t)$

$\rightarrow Terminates(Walk(l_2), At(Monkey, l_1), t)$

$Happens(Walk(l), t)$

$\rightarrow \neg HoldsAt(At(Monkey, l), t) \wedge \neg HoldsAt(OnBox, t)$

$HoldsAt(HasBananas, t) \wedge HoldsAt(At(Monkey, l), t)$

$\rightarrow HoldsAt(At(Bananas, l), t)$

....

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

We need to minimize the extents of *Initiates*, *Terminates*, *Happens*.

# Domain Independent Event Calculus Axioms

$Happens(e, t) \wedge Initiates(e, f, t) \rightarrow HoldsAt(f, t+1)$

$HoldsAt(f, t) \wedge \neg ReleasedAt(f, t+1) \wedge$   
 $\neg \exists e (Happens(e, t) \wedge Terminates(e, f, t)) \rightarrow HoldsAt(f, t+1)$

....

# Circumscription in Event Calculus

In order to minimize the extents of *Initiates*, *Terminates*, *Releases*, *Happens*.

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

$$\text{CIRC}[F; \mathbf{p}] = F \wedge (\text{2nd-order formula that makes } \mathbf{p} \text{ minimal})$$

# Event Calculus Domain Description

An event calculus domain description is defined as

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

where

- ▶  $\Sigma$  is a conjunction of “effect” axioms
  - ▶  $[\textit{condition}] \rightarrow \textit{Initiates}(e, f, t)$
  - ▶  $[\textit{condition}] \rightarrow \textit{Terminates}(e, f, t)$
  - ▶  $[\textit{condition}] \rightarrow \textit{Releases}(e, f, t)$
- ▶  $\Delta$  is a conjunction of “event occurrence” axioms
  - ▶  $\textit{Happens}(e, t)$
- ▶  $F$  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>

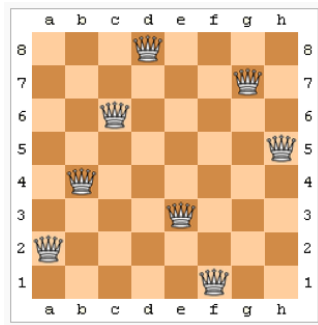


# What is Answer Set Programming (ASP)?

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.

# ASP Example: 8-Queens Problem



```
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.

1996: *S*MODELS: first answer set solver.

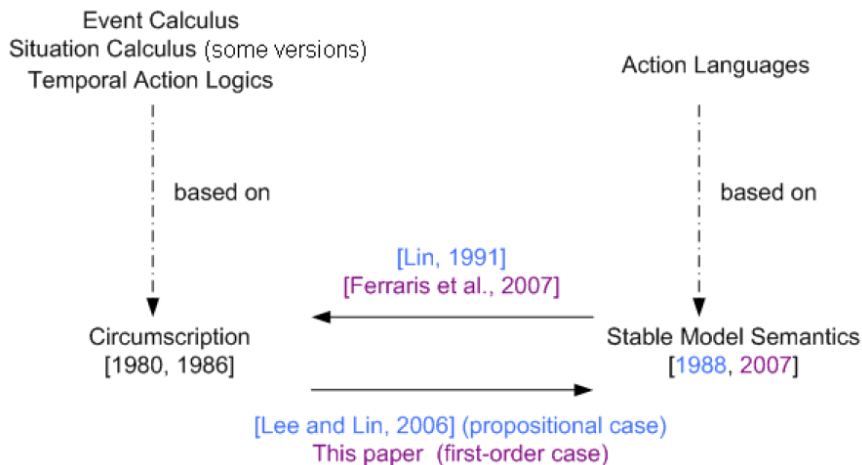
1999: ASP identified as a new programming paradigm.

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



# Turning Event Calculus Description into SM

(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

$$(HoldsAt(f, t) \wedge \neg ReleasedAt(f, t+1) \wedge \neg \exists e (Happens(e, t) \wedge Terminates(e, f, t))) \rightarrow HoldsAt(f, t+1).$$

is turned into the conjunction of

$$\begin{aligned} & (HoldsAt(f, t) \wedge \neg ReleasedAt(f, t+1) \wedge \neg q(f, t)) \rightarrow HoldsAt(f, t+1) \\ & Happens(e, t) \wedge Terminates(e, f, t) \rightarrow q(f, t) \end{aligned}$$

and then turned into rules

$$\begin{aligned} HoldsAt(f, t+1) & \leftarrow HoldsAt(f, t), \text{not } ReleasedAt(f, t+1), \\ & \text{not } q(f, t) \\ q(f, t) & \leftarrow Happens(e, t), Terminates(e, f, t) \end{aligned}$$

## 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)

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

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)

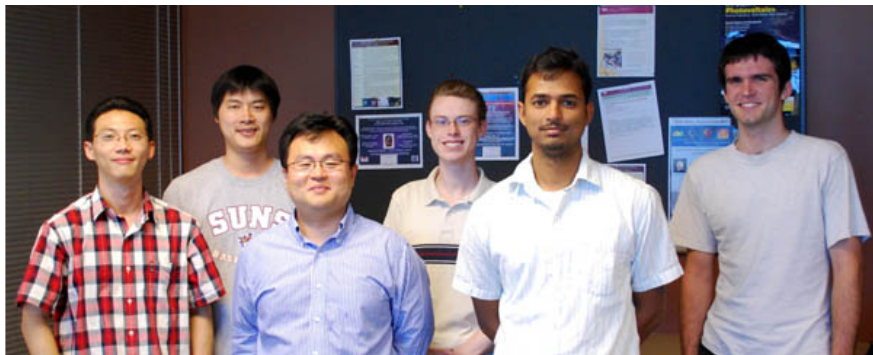
Problem (max. time)	DEC reasoner	ECASP w/ LPARSE + CMODELS	ECASP w/ GRINGO + CLASP	ECASP w/ CLINGO
Falling w/ AntiTraj (15)	270.2 (269.3+0.9) A:416/C:3056	0.74 (0.66+0.08) A:5757/R:10480/C:0	0.10 (0.08+0.02) A:4121/R:7820	0.08
Falling w/ Events (25)	107.70 (107.50+0.20) A:1092/C:12351	34.77 (30.99+3.78) A:1197/R:390319/C:1393	2.90 (2.01+0.89) A:139995/R:208282	2.32
HotAir Baloon (15)	61.10 (61.10+0.00) A:288/C:1163	0.19 (0.16+0.03) A:489/R:2958/C:678	0.04 (0.03+0.01) A:1137/R:1909	0.03
Telephone1 (40)	18.00 (17.50+0.50) A:5419/C:41750	1.70 (1.51+0.19) A:23978/R:30005/C:0	0.31 (0.26+0.05) A:21333/R:27201	0.25

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

# Conclusion

- ▶ 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