Course: Common Sense Reasoning

3. Event Calculus

Martin Molina

UNIVERSIDAD POLITECNICA MADRID

TECHNICA IMPENDI NATIONI

The event calculus is a logic-based language for reasoning about actions and their effects

- Different contributors:
  - Mueller, 2014
  - R.Kowalski, M.Sergot 1986
  - M.Shanahan, R.Miller, 1990, …
- Includes common sense issues:
  - Default reasoning
  - Indirect effects
  - Continuous change
  - Law of inertia
  - Others (delayed effects, concurrent events, etc.)

The representation is based on many-sorted first order logic

Sorts:

e: event

f: fluent

t: timepoint

Predicates:

$Happens(e, t)$

$HoldsAt(f, t)$

$ReleasedAt(f, t)$

$Initiates(e, f, t)$

$Terminates(e, f, t)$

$Releases(e, f, t)$

…

Axioms:

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

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

$$Happens(e, t) \wedge Releases(e, f, t) \rightarrow ReleasedAt(f, t + 1)$$

…

Example: Wake up

Sorts:

$a: agent$

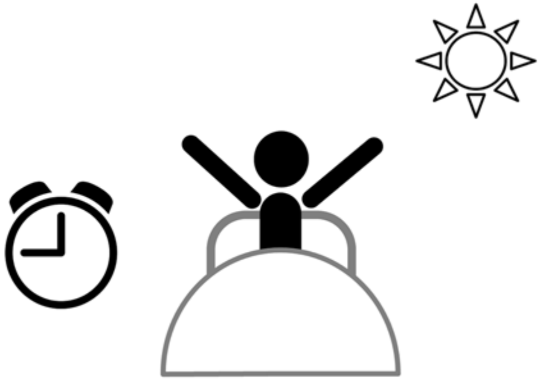
Events:

$WakeUp(a)$

$FallSleep(a)$

Fluents:

$Awake(a)$



Domain specific axioms:

$Initiates(WakeUp(a), Awake(a), t)$
$Terminates(FallAsleep(a), Awake(a), t)$

Premises:

$John: agent$

$\neg HoldsAt(Awake(John), 0)$

$Happens(WakeUp(John), 1)$

Question:

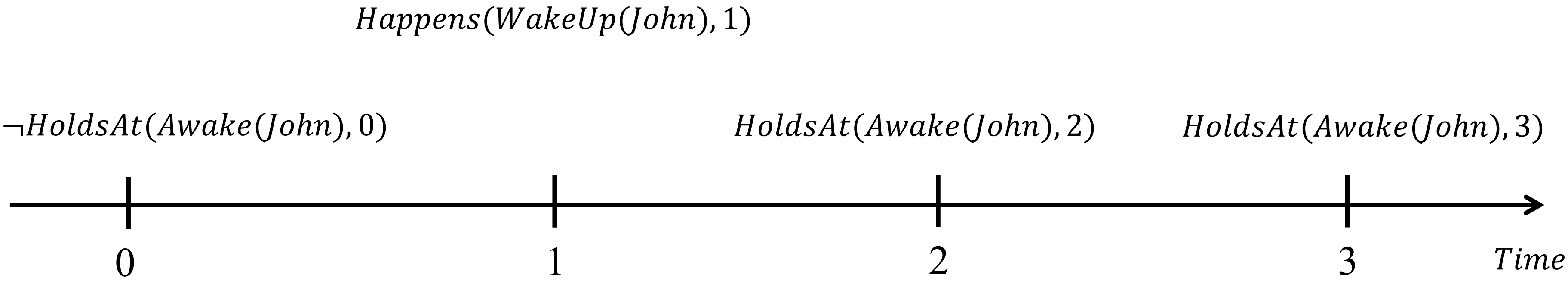
$HoldsAt(Awake(John), 3)$?

Axioms:

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

$Initiates(WakeUp(a), Awake(a), t)$

$Terminates(FallAsleep(a), Awake(a), t)$



$Happens(WakeUp(John), 1)$

$\neg HoldsAt(Awake(John), 0)$

$HoldsAt(Awake(John), 2)$

$HoldsAt(Awake(John), 3)$

Event calculus uses circumscription

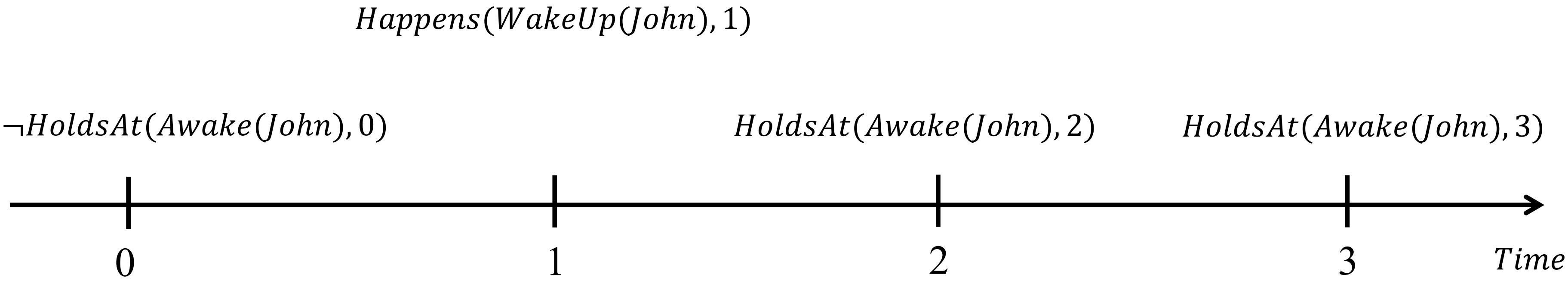
The “only known” event is that John wakes up at timepoint 1:

$$Happens(WakeUp(John), 1)$$

We assume that there is no other event happening, for example:

$$Happens(WakeUp(John), 2) = FALSE$$

We consider $Happens(e, t)$ as false as possible (Circumscription)



$Happens(WakeUp(John), 1)$

$\neg HoldsAt(Awake(John), 0)$

$HoldsAt(Awake(John), 2)$

$HoldsAt(Awake(John), 3)$

Circumscription of $\boldsymbol{Happens(e,t)}$

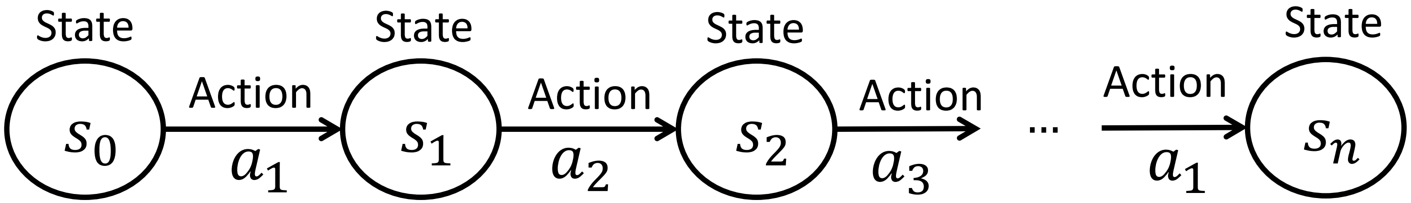
The “only known” event is that John wakes up at timepoint 1:

$$Happens(WakeUp(John), 1)$$

Circumscription of $Happens(e,t)$ means logically:

$$Happens(e,t) \leftrightarrow (e = WakeUp(John) \wedge t = 1)$$

Event calculus admits different types of reasoning



$$s_0 + \{a_1, a_2, \dots, a_n\} = s_n$$

- Deduction (i.e., temporal projection)

  Given s_0 and $\{a_1, a_2, \dots, a_n\}$, determine s_n

- Postdiction

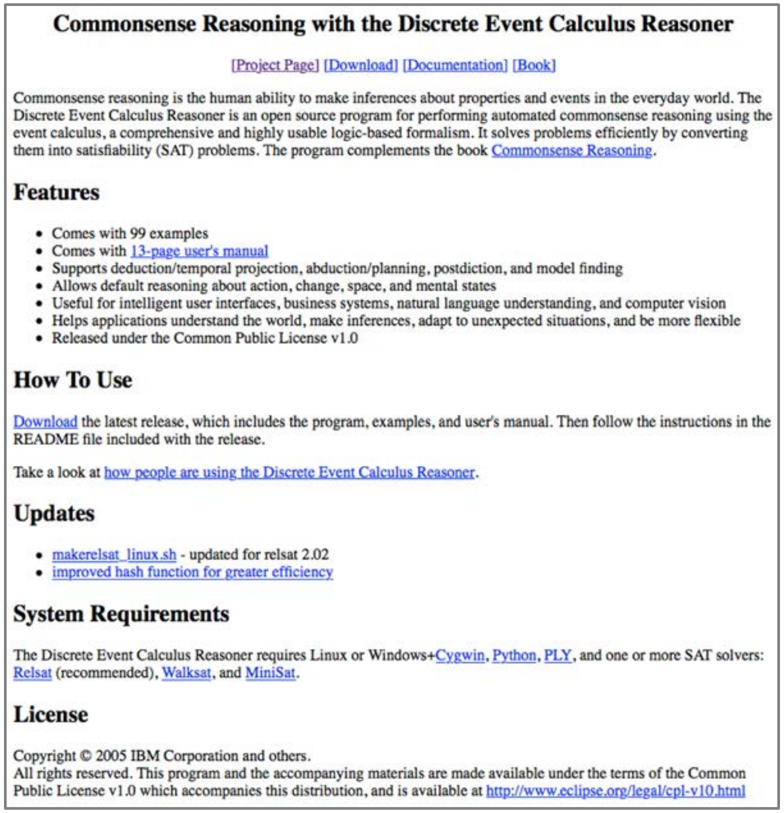
  Given s_n and $\{a_1, a_2, \dots, a_n\}$, determine s_0

- Abduction (i.e., planning)

  Given s_0 and s_n, determine $\{a_1, a_2, \dots, a_n\}$

“Discrete Event Calculus Reasoner” is a program for event calculus

- The program provides a logic language and inference [Mueller, 2014]
- The user writes the domain knowledge base
- It is publicly free available



Commonsense Reasoning with the Discrete Event Calculus Reasoner

[Project Page] [Download] [Documentation] [Book]

Commonsense reasoning is the human ability to make inferences about properties and events in the everyday world. The Discrete Event Calculus Reasoner is an open source program for performing automated commonsense reasoning using the event calculus, a comprehensive and highly usable logic-based formalism. It solves problems efficiently by converting them into satisfiability (SAT) problems. The program complements the book Commonsense Reasoning.

Features

- Comes with 99 examples
- Comes with 13-page user's manual
- Supports deduction/temporal projection, abduction/planning, postdiction, and model finding
- Allows default reasoning about action, change, space, and mental states
- Useful for intelligent user interfaces, business systems, natural language understanding, and computer vision
- Helps applications understand the world, make inferences, adapt to unexpected situations, and be more flexible
- Released under the Common Public License v1.0

How To Use

Download the latest release, which includes the program, examples, and user's manual. Then follow the instructions in the README file included with the release.

Take a look at how people are using the Discrete Event Calculus Reasoner.

Updates

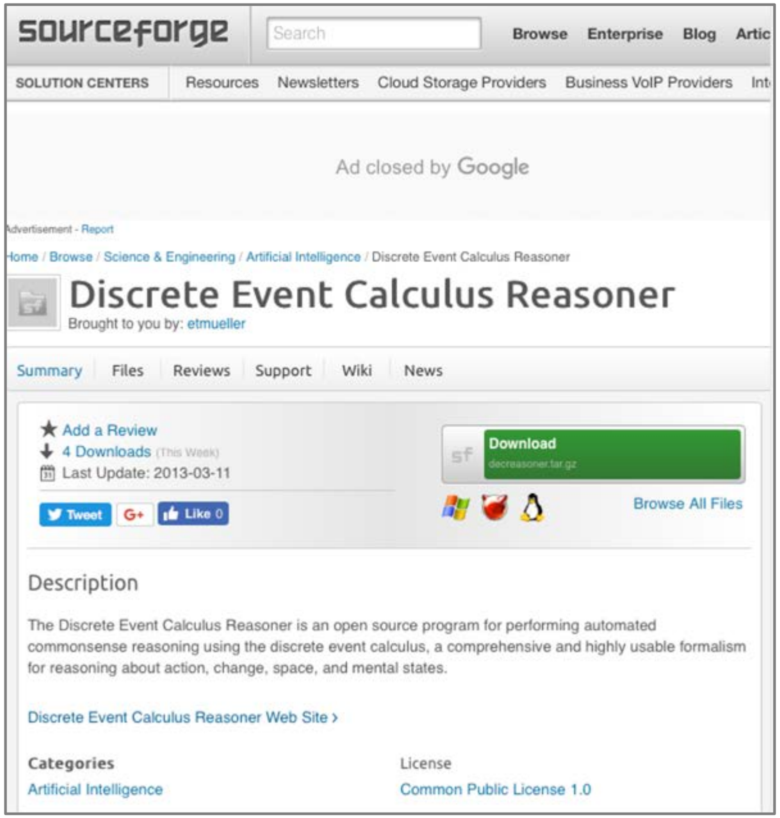
- makerelsat_linux.sh - updated for relsat 2.02
- improved hash function for greater efficiency

System Requirements

The Discrete Event Calculus Reasoner requires Linux or Windows+Cygwin, Python, PLY, and one or more SAT solvers: Relsat (recommended), Walksat, and MiniSat.

License

Copyright © 2005 IBM Corporation and others.
All rights reserved. This program and the accompanying materials are made available under the terms of the Common Public License v1.0 which accompanies this distribution, and is available at http://www.eclipse.org/legal/cpl-v10.html



Ad closed by Google

Advertisement - Report

Home / Browse / Science & Engineering / Artificial Intelligence / Discrete Event Calculus Reasoner

Discrete Event Calculus Reasoner

Brought to you by: etmueller

Last Update: 2013-03-11

Description

The Discrete Event Calculus Reasoner is an open source program for performing automated commonsense reasoning using the discrete event calculus, a comprehensive and highly usable formalism for reasoning about action, change, space, and mental states.

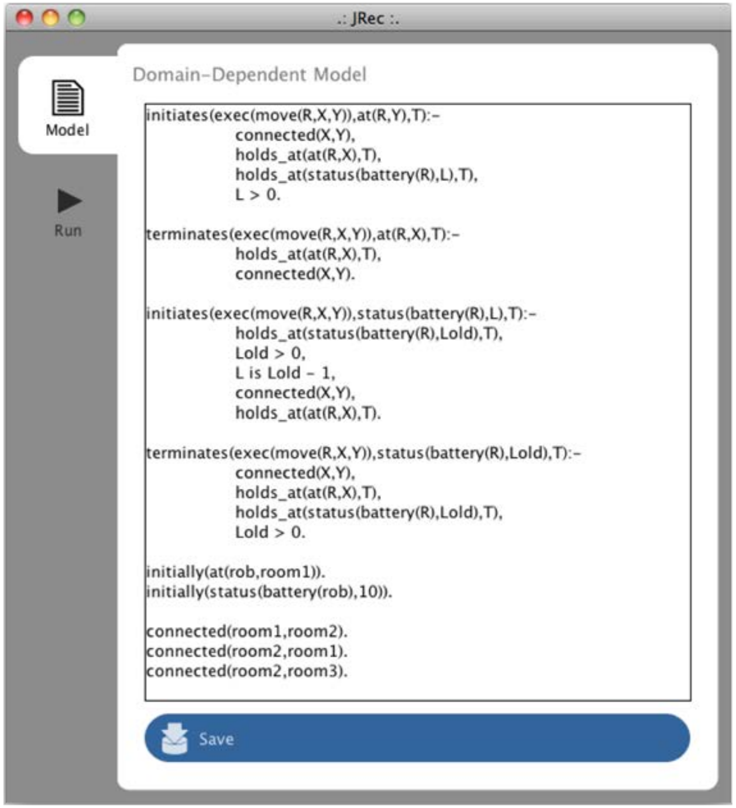
Discrete Event Calculus Reasoner Web Site ›

Categories: Artificial Intelligence

License: Common Public License 1.0

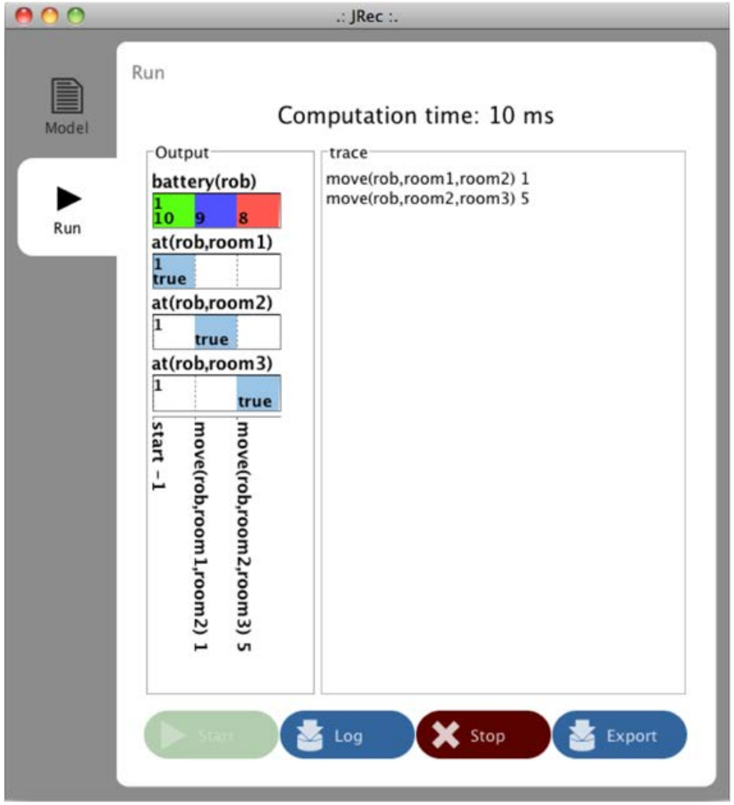
Some extensions for event calculus have been proposed to improve computational efficiency

- Reactive event calculus [Chesani et al., 2010; Bragaglia et al., 2012]
- Extensions of event calculus [Cervesato et al. 2000]
- Cached event calculus [Chittaro, Montanari,1996]
- JREC: `https://www.inf.unibz.it/~montali/tools.html`



Domain-Dependent Model

```
initiates(exec(move(R,X,Y)),at(R,Y),T):-
        connected(X,Y),
        holds_at(at(R,X),T),
        holds_at(status(battery(R),L),T),
        L > 0.

terminates(exec(move(R,X,Y)),at(R,X),T):-
        holds_at(at(R,X),T),
        connected(X,Y).

initiates(exec(move(R,X,Y)),status(battery(R),L),T):-
        holds_at(status(battery(R),Lold),T),
        Lold > 0,
        L is Lold - 1,
        connected(X,Y),
        holds_at(at(R,X),T).

terminates(exec(move(R,X,Y)),status(battery(R),Lold),T):-
        connected(X,Y),
        holds_at(at(R,X),T),
        holds_at(status(battery(R),Lold),T),
        Lold > 0.

initially(at(rob,room1)).
initially(status(battery(rob),10)).

connected(room1,room2).
connected(room2,room1).
connected(room2,room3).
```



Run

Computation time: 10 ms

trace

```
move(rob,room1,room2) 1
move(rob,room2,room3) 5
```

Course “Common sense reasoning”.
© 2019 Martin Molina

This work is licensed under Creative Commons license CC BY-NC-SA 4.0:
https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

Work citation in APA style:

> Molina, M. (2019). Common sense reasoning [Lecture slides]. OpenCourseWare, Universidad Politécnica de Madrid. Retrieved from http://ocw.upm.es/course