A complete reasoning procedure in the presence of incomplete information

Presented by Tu Phan

Outline

- Reasoning under Incomplete Information
  - Possible World Semantics vs Approximation Semantics
- An alternative approach
- Application to Conformant Planning
- Related work
- Conclusion

A running example

- Bomb-in-the-toilet domain
  - one package, one toilet
  - the package may or may not contain a bomb
  - dunking the package causes the bomb to be disarmed if it is armed
  - flushing the toilet makes it unclogged
- **Question:**
  - Whether or not the bomb is disarmed after flushing the toilet and then dunking the package

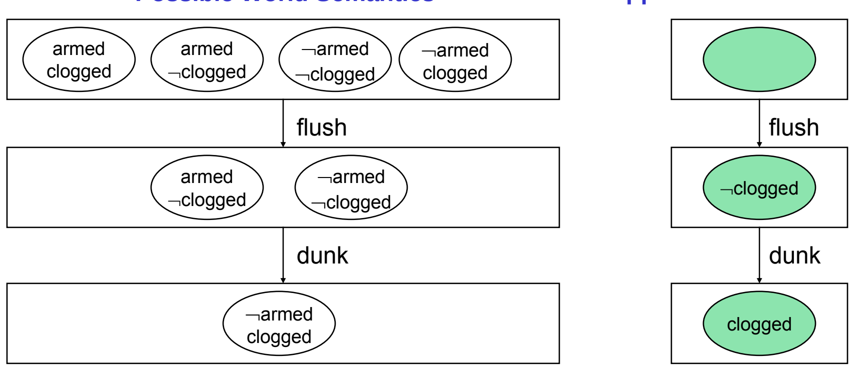
- **Domain Description D_1** (in language A)
  - **executable** dunk **if** ¬clogged
  - dunk **causes** ¬armed **if** armed
  - dunk **causes** clogged
  - flush **causes** ¬clogged
- **Initial condition:** $I_1 = \varnothing$
- **Question:** $(D_1,I_1) \models \neg armed$ after [flush;dunk]?

3

Reasoning under Incomplete Information

Possible World Semantics vs. Approximation Semantics

Possible World Semantics | **Approximation Semantics**



$(D_1,I_1) \models_P \neg clogged$ **after** [flush]
$(D_1,I_1) \models_P \neg armed$ **after** [flush;dunk]

$(D_1,I_1) \models_a \neg clogged$ **after** [flush]
(D_1,I_1) **not** $\models_a \neg armed$ **after** [flush;dunk]

4

Reasoning under Incomplete Information

Possible World Semantics vs Approximation Semantics

- Possible World Semantics
  - more answers
    - $(D_1,I_1) \models_p \neg$armed **after** [flush;dunk]
  - less efficient: at any time, number of possible worlds may be very big
- Approximation Semantics
  - less answers:
    - (D_1,I_1) not $\models_a \neg$armed **after** [flush;dunk]
  - more efficient: consider only one partial state at a time
  - sound but incomplete w.r.t. possible world

5

Alternative Approach

- Based on the approximation semantics
- In the beginning, consider a set of partial states rather than a single one
  - done by partitioning over the truth values of some unknown fluents

6

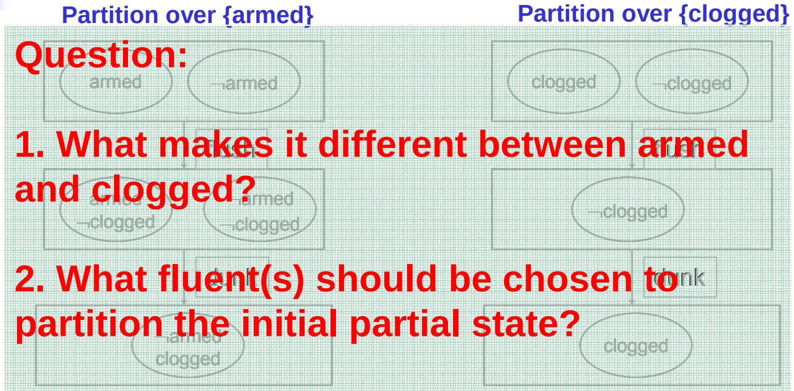
Alternative Approach

Partition over {armed} | Partition over {clogged}

Question:

1. What makes it different between armed and clogged?

2. What fluent(s) should be chosen to partition the initial partial state?



$(D_1,I_1) \models_A \neg$clogged **after** [flush]
$(D_1,I_1) \models_A \neg$armed **after** [flush;dunk]

$(D_1,I_1) \models_A \neg$clogged **after** [flush]
(D_1,I_1) **not** $\models_A \neg$armed **after** [flush;dunk]

Decisive sets of fluents

- A set of unknown fluents is *decisive* if it can be used to partition the initial partial state in order for $\models_A$ to be complete
- E.g.:
  - {armed},{armed,clogged} are decisive
  - {clogged} is not decisive

Algorithm for computing a decisive set of fluents

- Objectives
  - computationally efficient
  - returned decisive set should be as small as possible
    - help reduce the search space
- Method
  - Based on the concept of dependencies

9

Dependencies

- A literal l *depends* on a literal l_1 if either
  - $l = l_1$
  - $\neg l$ depends on $\neg l_1$
  - there exists **a causes l if p** s.t. $l_1 \in p$, or
  - there exists l_2 such that l depends on l_2 and l_2 depends on l_1

- **Domain**
  - **executable** dunk **if** ¬clogged
  - dunk **causes** ¬armed **if** armed
  - dunk **causes** clogged
  - flush **causes** ¬clogged
- **Dependencies**
  - Ω(armed) = {armed, ¬armed}
  - Ω(clogged) = {clogged}

10

Computing a decisive set of fluents

- Decisive(D,I)
  - Initialize $F = \emptyset$
  - Compute the dependency relationship
  - For each unknown fluent f
    - if there exists l s.t. l depends on both f and $\neg$f then
      - $F = F \cup \{f\}$
  - return F

- Theorem:
  - Decisive(D,I) is a decisive set of fluents, provided that every action has at most one executability condition

Example

- **Domain D_1**
  - **executable** dunk **if** $\neg$clogged
  - dunk **causes** $\neg$armed **if** armed
  - dunk **causes** clogged
  - flush **causes** $\neg$clogged

Initial Partial State: $I_1 = \emptyset$

- **Dependencies**
  - Ω(armed) = {armed, $\neg$armed}
  - Ω($\neg$armed) = {$\neg$armed,armed}

  Ω(clogged) = {clogged}
  Ω($\neg$clogged) = {$\neg$clogged}

- DECISIVE(D_1,I_1) = {armed}
  - armed depends on both armed and $\neg$armed
  - no literal l such that l depends on both clogged and $\neg$clogged

New approach properties

- Sound and complete
- More compact than possible world semantics in many cases
- Computing a decisive set of fluents can be done in polynomial time

13

What are missing?

- What if we have more than one executability condition for an action
  - Solution: Dependencies between actions and literals
- What if we have more than one initial partial state, for example
  initially f | g
  - Solution:
    - For each partial state, partition it using the same procedure for computing a decisive set
- Static causal laws:
  - At present, we only have the result for domains with static causal laws whose body contains at most one literal

14

Application – Conformant Planning

- Problem
  - Given: an action theory (D,I), and a set G of literals
  - Find: a sequence of actions that, when executed in any possible initial state, always achieves G
- Our approach
  - If the goal is {-clogged} then do not need to partition I
  - Modify algorithm DECISIVE(D,I) so as to take

Computing a decisive set of fluents

- Decisive(D,I,G)
  - Initialize F = $\emptyset$
  - Compute the dependency relationship
  - For each unknown fluent f
    - if there exists l $\in$ G s.t. l depends on both f and $\neg$f then
      - F = F $\cup$ {f}
  - return F
- Theorem:
  - Decisive(D,I,G) is a decisive set of fluents for planning problem (D,I,G)

Computing a decisive set
Example

- **Domain D_1**
  - **executable** dunk **if** ¬clogged
  - dunk **causes** ¬armed **if** armed
  - dunk **causes** clogged
  - flush **causes** ¬clogged

Initial Partial State: $I_1 = \emptyset$

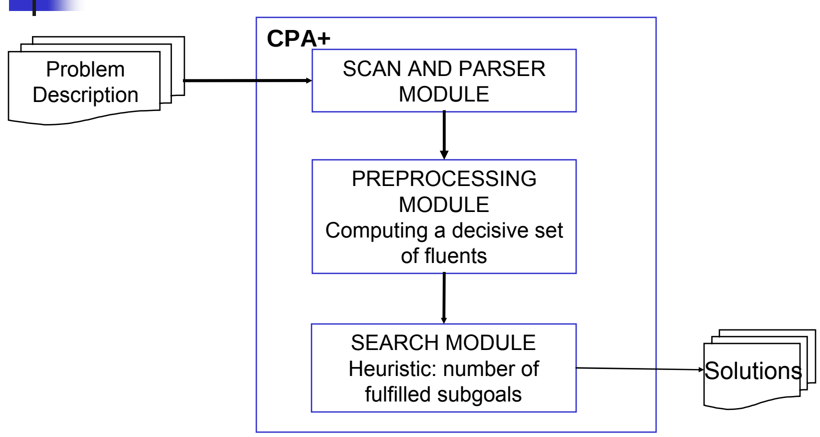
- **Dependencies**
  - Ω(armed) = {armed, ¬armed}
  - Ω(¬armed) = {¬armed,armed}

  Ω(clogged) = {clogged}
  Ω(¬clogged) = {¬clogged}

- DECISIVE(D_1,I_1,{clogged}) = $\emptyset$
  - no fluent f such that clogged depends on both f and ¬f
- DECISIVE(D_1,I_1,{¬armed}) = {armed}
  - ¬armed depends on both armed and ¬armed

17

CPA+ - A Conformant Planner



18

CPA+ - Performance

Problem	KACMBP		CFF		POND		CPA*		CPA+	
	PL	Time	PL	Time	PL	Time	PL	Time	PL	Time
bomb(10,1)	19	0.01	19	0.05	19	2.61	19	1.519 (0.06)	19	0.009 (0.002)
bomb(50,1)	99	0.51	99	5.33		AB		TO	99	0.506 (0.169)
bomb(100,1)	199	3.89	199	121.8		AB		TO	199	3.758 (1.322)
bomb(10,5)	15	0.09	15	0.07		AB	15	6.006 (0.07)	15	0.026 (0.005)
bomb(50,5)	95	1.66	95	4.7		AB		TO	95	1.054 (0.196)
bomb(100,5)	195	6.92	195	113.95		AB		TO	195	6.805 (1.41)
bomb(10,10)	10	0.3	10	0.05		AB	10	15.003 (0.079)	10	0.048 (0.008)
bomb(50,10)	90	5.39	90	4.04		AB		TO	90	1.92 (0.232)
bomb(100,10)	190	35.83	190	102.56		AB		TO	190	11.096 (1.517)

Bomb in the toilet domain

CPA*: using possible world semantics
CPA+: using the new approach
PL: Plan length; TO: Time out; AB: Abnormal Termination
Times are in seconds

19

Related Work

- Approximations
  - [Son & Chitta, AI 2001]
  - [Son, Tu & Chitta, LPNMR 2004]
  - [Son, Tu, Gelfond & Ricardo, AAAI 2005]
  - [Son, Tu, Gelfond & Ricardo, LPNMR 2005]
- Irrelevant Information
  - [Nebel, Dimopoulos & Koehler, ECP 1997]
- Reduced sets of actions
  - [Haslum & Johnsson, ICAPS 2000]
- Isolated sets of actions and fluents
  - [Lifschitz & Ren 2004]

20

Conclusion

- New Approach
  - Sound and Complete
  - Efficient
- Application to conformant planning
  - A conformant planner competitive with other state-of-the-art planners