The AI Universe of "Actions": Agency, Causality, Commonsense and Deception

Chitta Baral and Tran Cao Son

Arizona State University and New Mexico State University

IJCAI 2019

1. Overview: a brief history of action languages **(20 mnt - Chitta)**
2. Action languages in single agent environments **(70 mnt - Son)**
   - The action language $\mathcal{A}$, state, and transition function
   - $\mathcal{AL}$: $\mathcal{A}+$ static causal laws, non-deterministic and sensing actions
   - GOLOG
   - Action languages: related approaches and planning
3. Action languages and causality **(30 mnt - Chitta)**
   - Pearl's do-calculus
4. Action languages in multi-agent environments **(60 mnt – Son)**
   - mA*, Kripke structure, update models, and transition function
   - mA* in epistemic planning
5. Action languages in commonsense reasoning **(18 mnt - Chitta)**
   - Commonsense reasoning and actions
   - Acquiring knowledge about actions
6. Open challenges, conclusion, and discussion **(12 mnt - Chitta)**

Actions in AI-literature

It starts with John McCarthy

i-trofimov.narod.ru/1959-McCarthy-Programs_with_Common_Sense.pdf

The argument the *advice taker* must produce in order to solve the problem deduces the following propositions in more or less the following order:

1. $at(I, desk) \rightarrow can(go(desk, car, walking))$
2. $at(I, car) \rightarrow can(go(home, airport, driving))$
3. $did(go(desk, car, walking)) \rightarrow at(I, car)$
4. $did(go(home, airport, driving)) \rightarrow at(I, airport)$
5. $canachult(at(I, desk), go(desk, car, walking), at(I, car))$
6. $canachult(at(I, car), go(home, airport, driving), at(I, airport))$
7. $canachult(at(I, desk), prog(go(desk, car, walking), go(home, airport, driving)) \rightarrow at(I, airport))$
8. $do(go(desk, car, walking))$

Actions in AI-literature

VERB PHYSICS: Relative Physical Knowledge of Actions and Objects

Maxwell Forbes **Yejin Choi**
Paul G. Allen School of Computer Science & Engineering
University of Washington
{mbforbes,yejin}@cs.washington.edu

Abstract

Learning commonsense knowledge from natural language text is nontrivial due to *reporting bias*: people rarely state the obvious, e.g., "My house is *bigger* than me." However, while rarely stated explicitly, this trivial everyday knowledge does influence the way people talk about the world, which provides indirect clues to reason about the world. For example, a statement like, "Tyler *entered* his house" implies that his house is *bigger* than Tyler.

In this paper, we present an approach to infer relative physical knowledge of actions and objects along five dimensions (e.g., size, weight, and strength) from unstructured natural language text. We frame knowledge acquisition as joint inference over two closely related problems: learning (1) relative physical knowledge of object pairs and (2) physical implications of actions when applied to those object pairs. Empirical results demonstrate that it is possible to extract knowledge of actions and objects from language and that joint

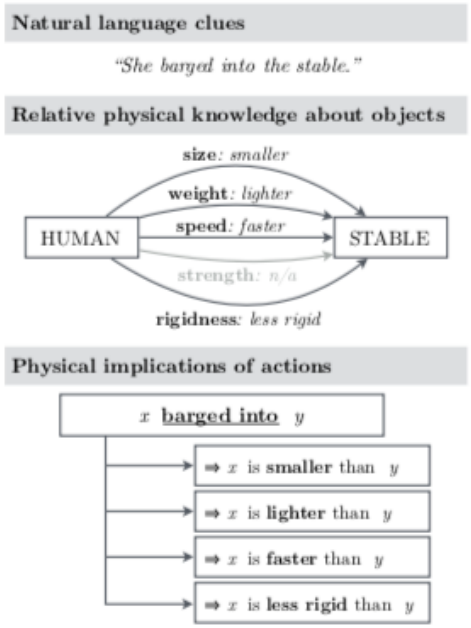


Figure 1: An overview of our approach. A verb's usage in language (top) implies physical relations between objects it takes as arguments. This allows us to reason about properties of specific objects (middle), as well as the knowledge implied by the

Actions in AI-literature

ATOMIC: An Atlas of Machine Commonsense for *If-Then* Reasoning

Maarten Sap[†⋆] **Ronan Le Bras**[†] **Emily Allaway**[⋆] **Chandra Bhagavatula**[†] **Nicholas Lourie**[†]
Hannah Rashkin[⋆] **Brendan Roof**[†] **Noah A. Smith**[†⋆] **Yejin Choi**[†⋆]
[⋆]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA
[†]Allen Institute for Artificial Intelligence, Seattle, USA
msap@cs.washington.edu

Abstract

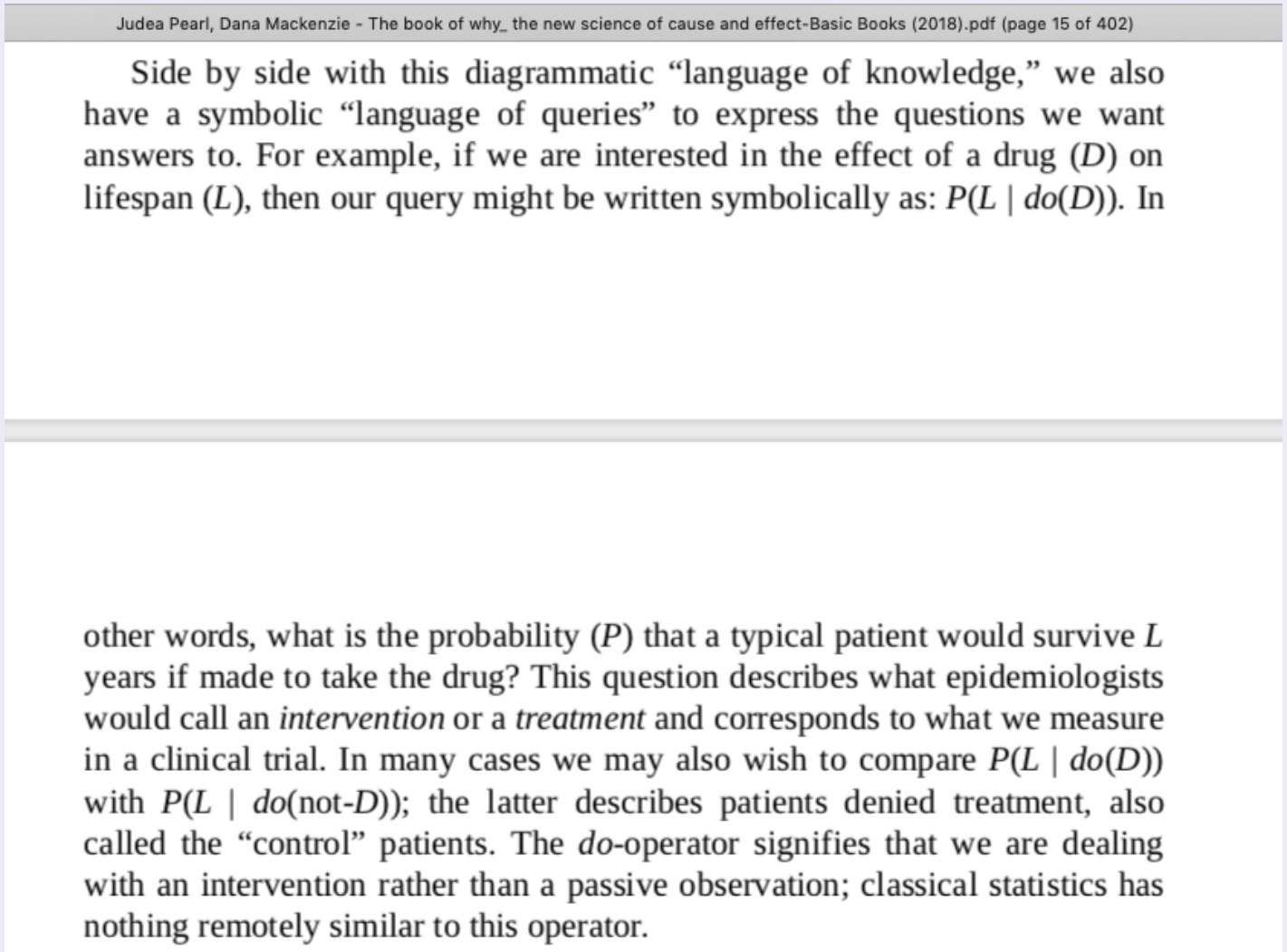
We present ATOMIC, an atlas of everyday commonsense reasoning, organized through 877k textual descriptions of inferential knowledge. Compared to existing resources that center around taxonomic knowledge, ATOMIC focuses on inferential knowledge organized as typed *if-then* relations with variables (e.g., "*if* X pays Y a compliment, *then* Y will likely return the compliment"). We propose nine *if-then* relation types to distinguish causes vs. effects, agents vs. themes, voluntary vs. involuntary events, and actions vs. mental states. By generatively training on the rich inferential knowledge described in ATOMIC, we show that neural models can acquire simple commonsense capabilities and reason about previously unseen events. Experimental results demonstrate that multitask models that incorporate the hierarchical structure of *if-then* relation types lead to more accurate inference compared to models trained in isolation, as measured by both automatic and human evaluation.

Introduction

Given a snapshot observation of an event, people can easily anticipate and reason about unobserved causes and effects in relation to the observed event: what might have happened just before, what might happen next as a result, and how different events are chained through causes and effects. For instance, if we observe an event "X repels Y's attack" (Fig-

Actions in AI-literature

book of why 1.png



Judea Pearl, Dana Mackenzie - The book of why_ the new science of cause and effect-Basic Books (2018).pdf (page 15 of 402)

Side by side with this diagrammatic "language of knowledge," we also have a symbolic "language of queries" to express the questions we want answers to. For example, if we are interested in the effect of a drug (D) on lifespan (L), then our query might be written symbolically as: $P(L \mid do(D))$. In

other words, what is the probability (P) that a typical patient would survive L years if made to take the drug? This question describes what epidemiologists would call an *intervention* or a *treatment* and corresponds to what we measure in a clinical trial. In many cases we may also wish to compare $P(L \mid do(D))$ with $P(L \mid do(\text{not-}D))$; the latter describes patients denied treatment, also called the "control" patients. The *do*-operator signifies that we are dealing with an intervention rather than a passive observation; classical statistics has nothing remotely similar to this operator.

Actions in AI-literature

book of why 2.png

Mathematically, we write the observed frequency of Lifespan L among patients who voluntarily take the drug as $P(L \mid D)$, which is the standard conditional probability used in statistical textbooks. This expression stands for the probability (P) of Lifespan L conditional on seeing the patient take Drug D. Note that $P(L \mid D)$ may be totally different from $P(L \mid do(D))$. This difference between seeing and doing is fundamental and explains why we do not regard the falling barometer to be a cause of the coming storm. Seeing the barometer fall increases the probability of the storm, while forcing it to fall does not affect this probability.

This confusion between seeing and doing has resulted in a fountain of paradoxes, some of which we will entertain in this book. A world devoid of $P(L \mid do(D))$ and governed solely by $P(L \mid D)$ would be a strange one indeed. For example, patients would avoid going to the doctor to reduce the probability of being seriously ill; cities would dismiss their firefighters to reduce the incidence of fires; doctors would recommend a drug to male and female patients but not to patients with undisclosed gender; and so on. It is hard to believe that less than three decades ago science did operate in such a world: the *do*-operator did not exist.

Actions in AI-literature

https://arxiv.org/pdf/1502.05698.pdf

Table 1: Sample statements and questions from tasks 1 to 10.

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 4: Two Argument Relations
The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Task 5: Three Argument Relations
Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill

Task 6: Yes/No Questions
John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

Task 7: Counting
Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

Task 8: Lists/Sets
Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
John took the apple.
What is Daniel holding? milk, football

Task 9: Simple Negation
Sandra travelled to the office.
Fred is no longer in the office.
Is Fred in the office? A:no
Is Sandra in the office? A:yes

Task 10: Indefinite Knowledge
John is either in the classroom or the playground.
Sandra is in the garden.
Is John in the classroom? A:maybe
Is John in the office? A:no

bABI 1.png

Actions in AI-literature

https://arxiv.org/pdf/1502.05698.pdf

Table 2: Sample statements and questions from tasks 11 to 20.

Task 11: Basic Coreference
Daniel was in the kitchen.
Then he went to the studio.
Sandra was in the office.
Where is Daniel? A:studio

Task 12: Conjunction
Mary and Jeff went to the kitchen.
Then Jeff went to the park.
Where is Mary? A: kitchen
Where is Jeff? A: park

Task 13: Compound Coreference
Daniel and Sandra journeyed to the office.
Then they went to the garden.
Sandra and John travelled to the kitchen.
After that they moved to the hallway.
Where is Daniel? A: garden

Task 14: Time Reasoning
In the afternoon Julie went to the park.
Yesterday Julie was at school.
Julie went to the cinema this evening.
Where did Julie go after the park? A:cinema
Where was Julie before the park? A:school

Task 15: Basic Deduction
Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? A:wolves

Task 16: Basic Induction
Lily is a swan.
Lily is white.
Bernhard is green.
Greg is a swan.
What color is Greg? A:white

Task 17: Positional Reasoning
The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square? A:yes
Is the red square to the left of the triangle? A:yes

Task 18: Size Reasoning
The football fits in the suitcase.
The suitcase fits in the cupboard.
The box is smaller than the football.
Will the box fit in the suitcase? A:yes
Will the cupboard fit in the box? A:no

Task 19: Path Finding
The kitchen is north of the hallway.
The bathroom is west of the bedroom.
The den is east of the hallway.
The office is south of the bedroom.
How do you go from den to kitchen? A: west, north
How do you go from office to bathroom? A: north, west

Task 20: Agent's Motivations
John is hungry.
John goes to the kitchen.
John grabbed the apple there.
Daniel is hungry.
Where does Daniel go? A:kitchen
Why did John go to the kitchen? A:hungry

bABI 2.png

Actions in AI-literature

and in the Winograd challenge

The need for thinking is perhaps even more evident in a much more difficult example, a variant of which was first presented by Terry Winograd (Winograd 1972), for whom we have named the schema:[2]

> The town councillors refused to give the angry demonstrators a permit because they feared violence. Who feared violence?
>
> Answer 0: the town councillors
> Answer 1: the angry demonstrators

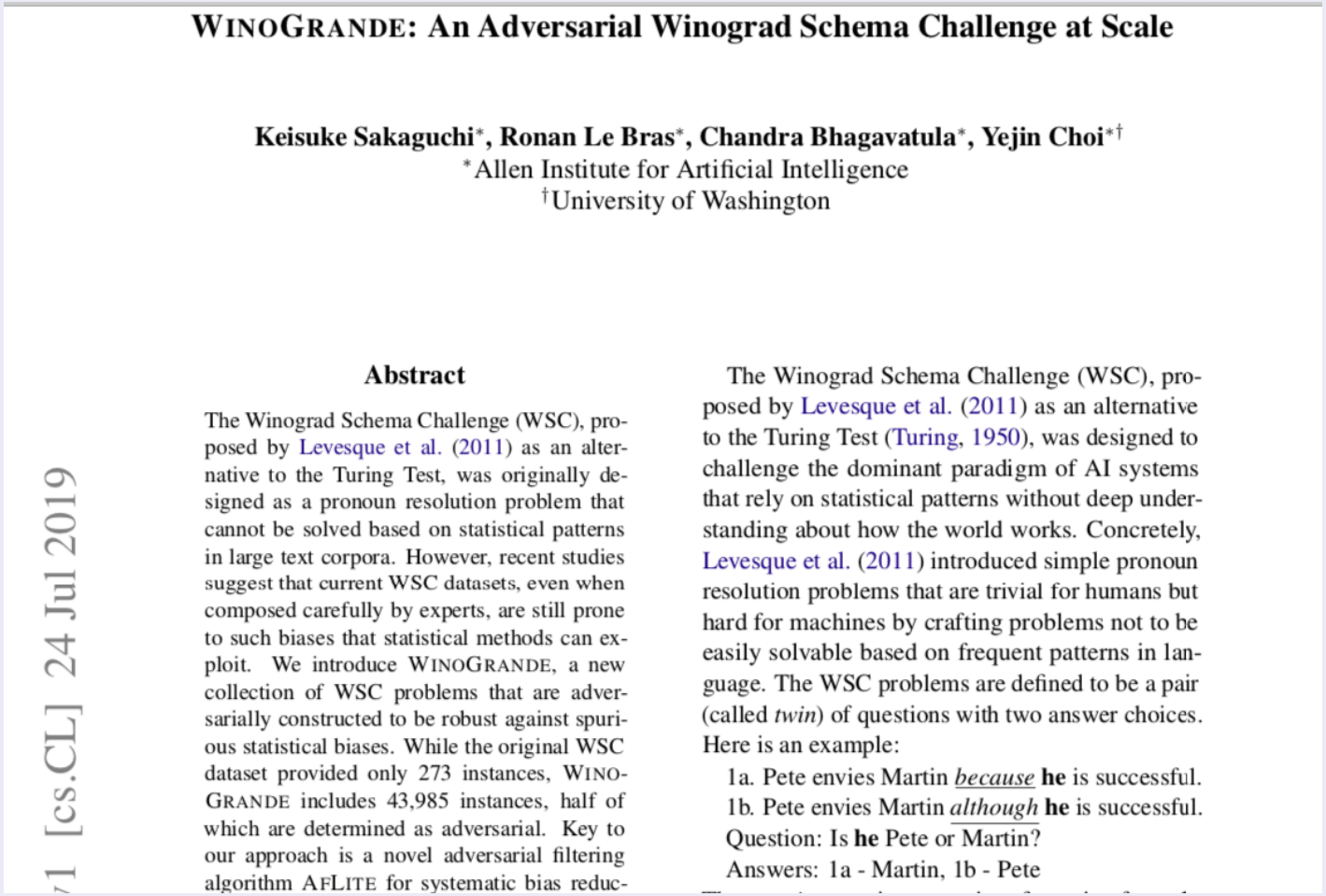
Here the special word is "feared" and its alternate is "advocated" as in the following:

> The town councillors refused to give the angry demonstrators a permit because they advocated violence. Who advocated violence?

KR Winograd.png

Actions in AI-literature

Winogrande.png



WINOGRANDE: An Adversarial Winograd Schema Challenge at Scale

Keisuke Sakaguchi*, Ronan Le Bras*, Chandra Bhagavatula*, Yejin Choi*†
*Allen Institute for Artificial Intelligence
†University of Washington

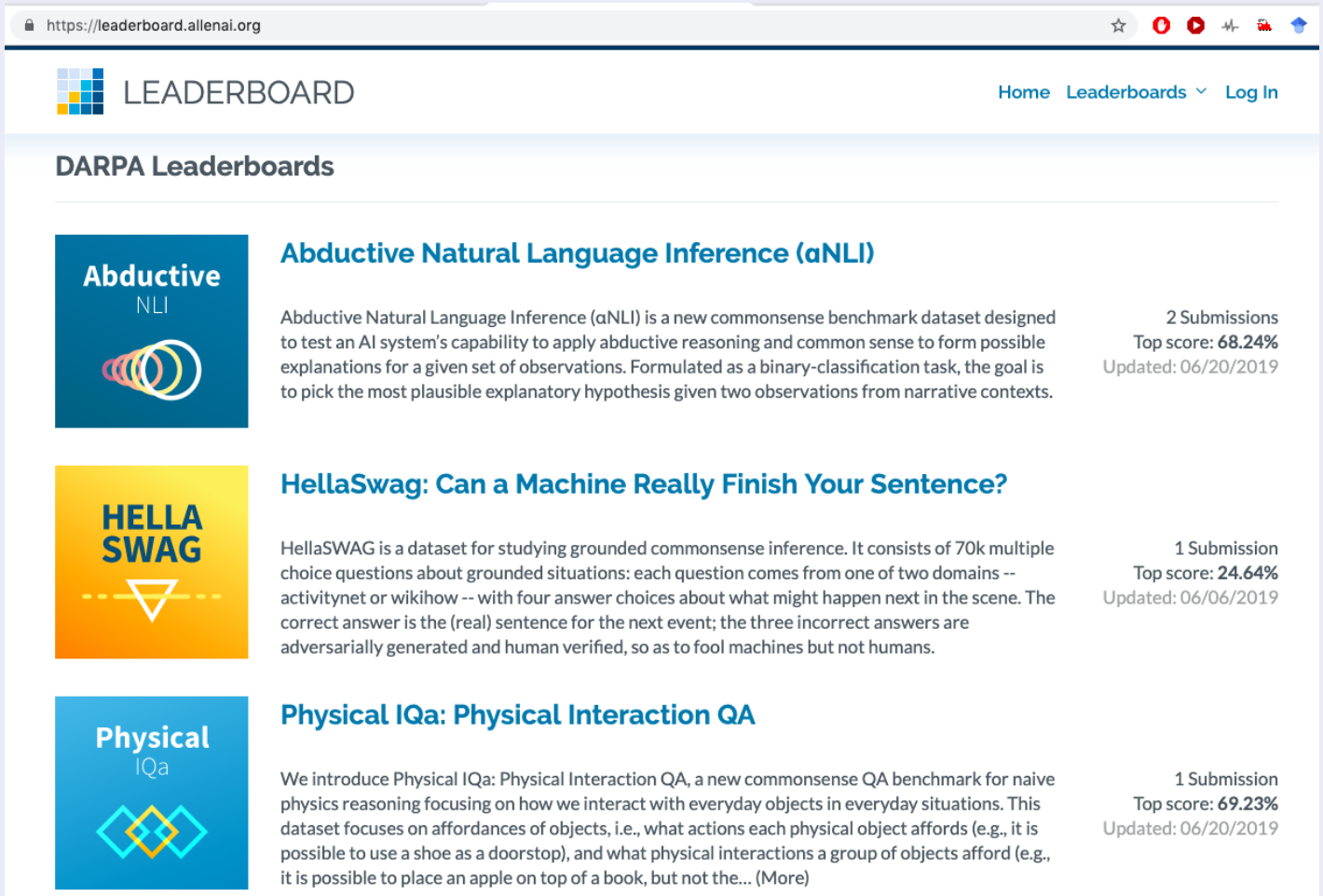
Abstract

The Winograd Schema Challenge (WSC), proposed by Levesque et al. (2011) as an alternative to the Turing Test, was originally designed as a pronoun resolution problem that cannot be solved based on statistical patterns in large text corpora. However, recent studies suggest that current WSC datasets, even when composed carefully by experts, are still prone to such biases that statistical methods can exploit. We introduce WINOGRANDE, a new collection of WSC problems that are adversarially constructed to be robust against spurious statistical biases. While the original WSC dataset provided only 273 instances, WINOGRANDE includes 43,985 instances, half of which are determined as adversarial. Key to our approach is a novel adversarial filtering algorithm AFLITE for systematic bias reduc-

The Winograd Schema Challenge (WSC), proposed by Levesque et al. (2011) as an alternative to the Turing Test (Turing, 1950), was designed to challenge the dominant paradigm of AI systems that rely on statistical patterns without deep understanding about how the world works. Concretely, Levesque et al. (2011) introduced simple pronoun resolution problems that are trivial for humans but hard for machines by crafting problems not to be easily solvable based on frequent patterns in language. The WSC problems are defined to be a pair (called *twin*) of questions with two answer choices. Here is an example:

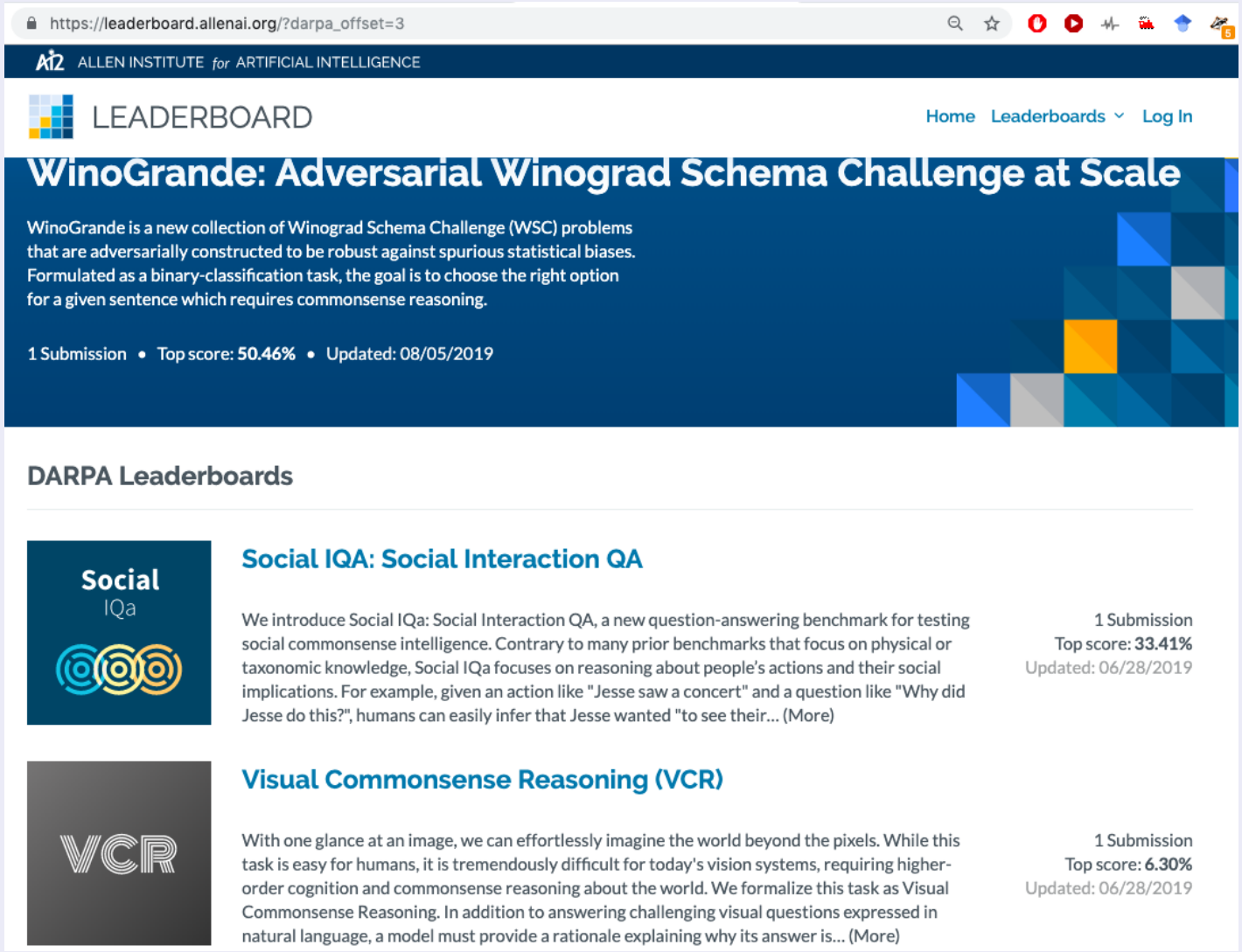
1a. Pete envies Martin *because* **he** is successful.
1b. Pete envies Martin *although* **he** is successful.
Question: Is **he** Pete or Martin?
Answers: 1a - Martin, 1b - Pete

Actions in AI-literature



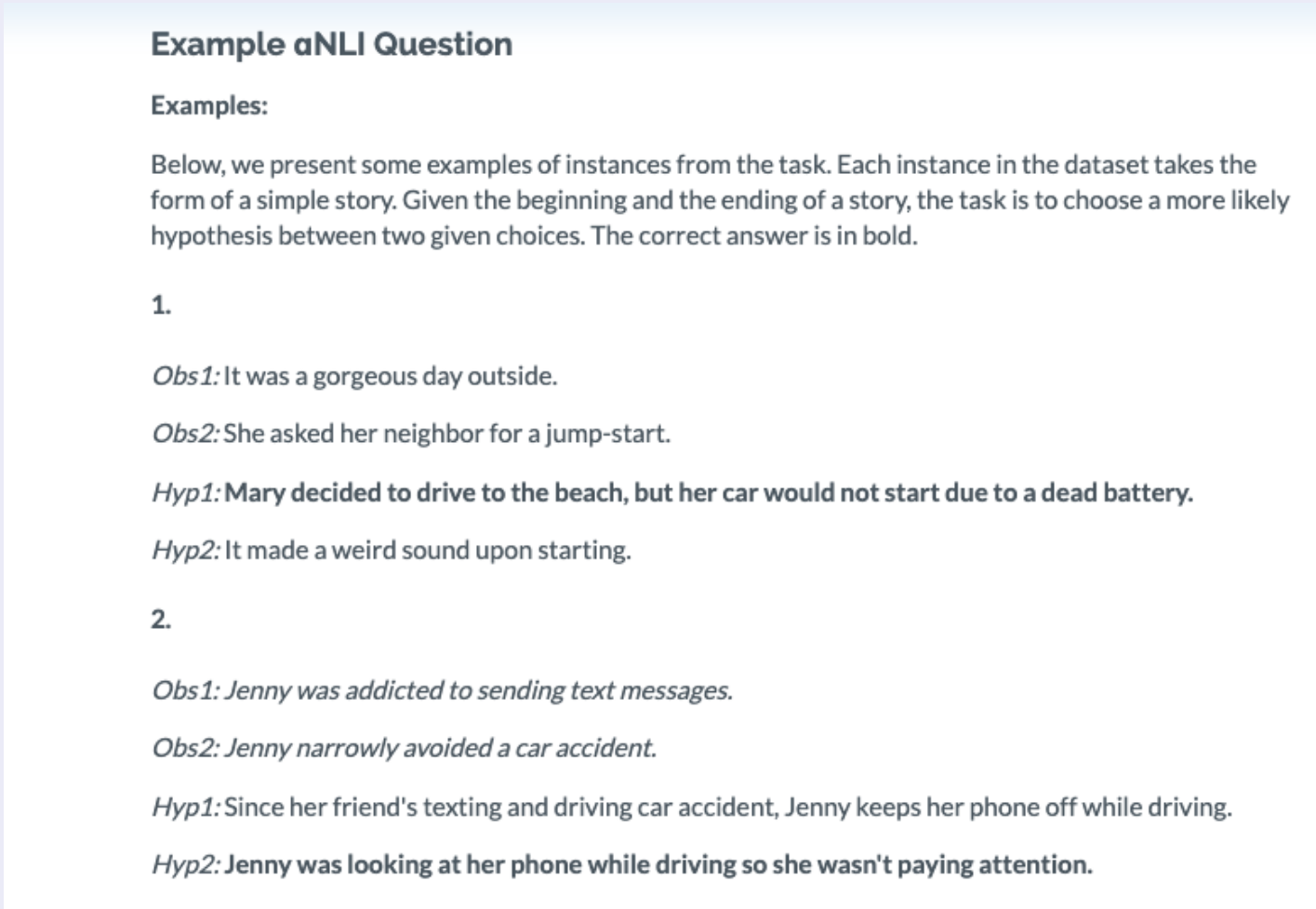
darpa mcs 1.png

Actions in AI-literature



darpa mcs 2.png

Actions in AI-literature



Example aNLI Question

Examples:

Below, we present some examples of instances from the task. Each instance in the dataset takes the form of a simple story. Given the beginning and the ending of a story, the task is to choose a more likely hypothesis between two given choices. The correct answer is in bold.

1.

Obs1: It was a gorgeous day outside.

Obs2: She asked her neighbor for a jump-start.

Hyp1: **Mary decided to drive to the beach, but her car would not start due to a dead battery.**

Hyp2: It made a weird sound upon starting.

2.

Obs1: Jenny was addicted to sending text messages.

Obs2: Jenny narrowly avoided a car accident.

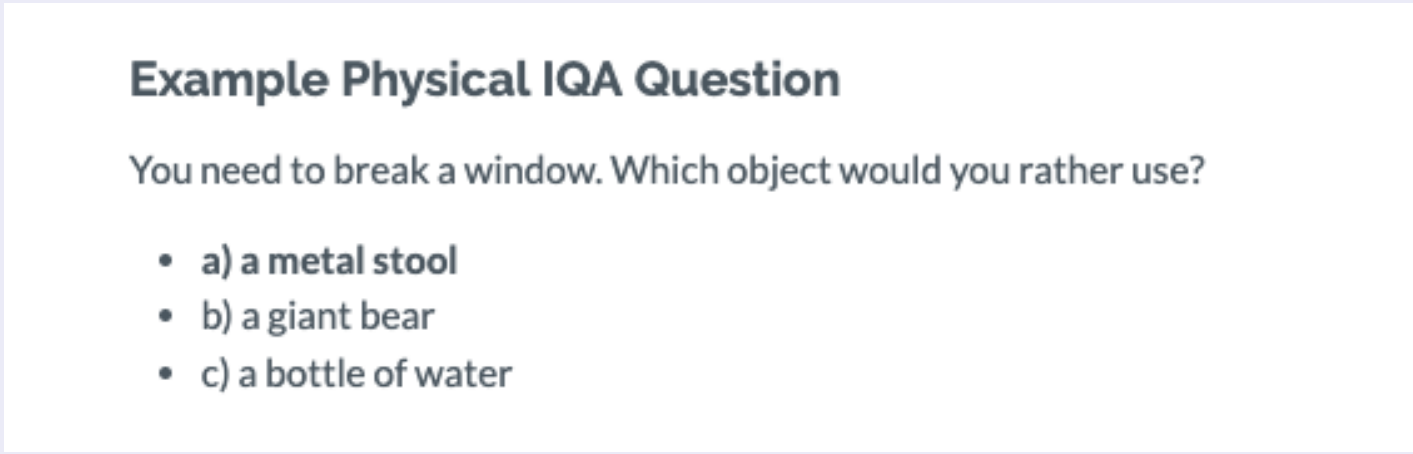
Hyp1: Since her friend's texting and driving car accident, Jenny keeps her phone off while driving.

Hyp2: **Jenny was looking at her phone while driving so she wasn't paying attention.**

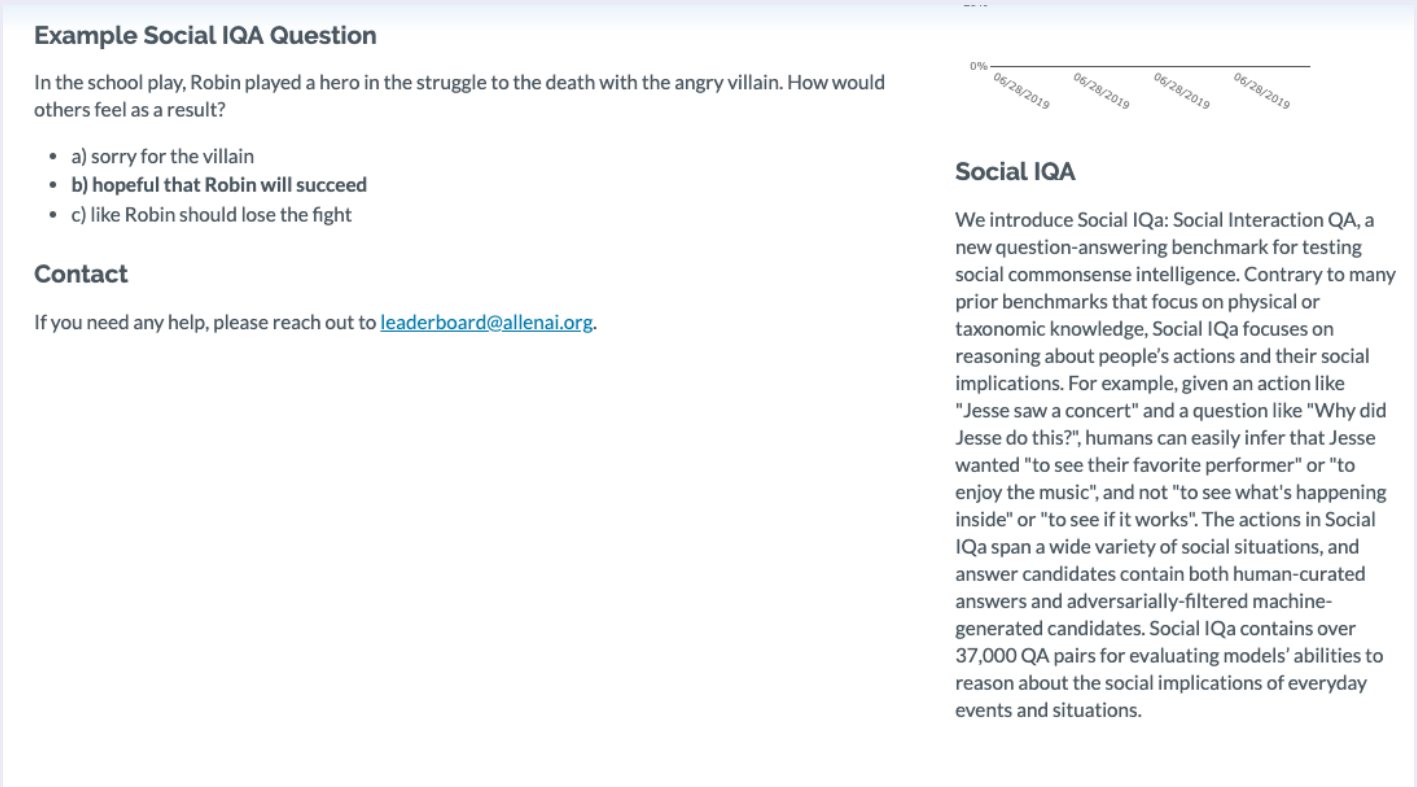
abductive nli.png

Actions in AI-literature

physical qa .png



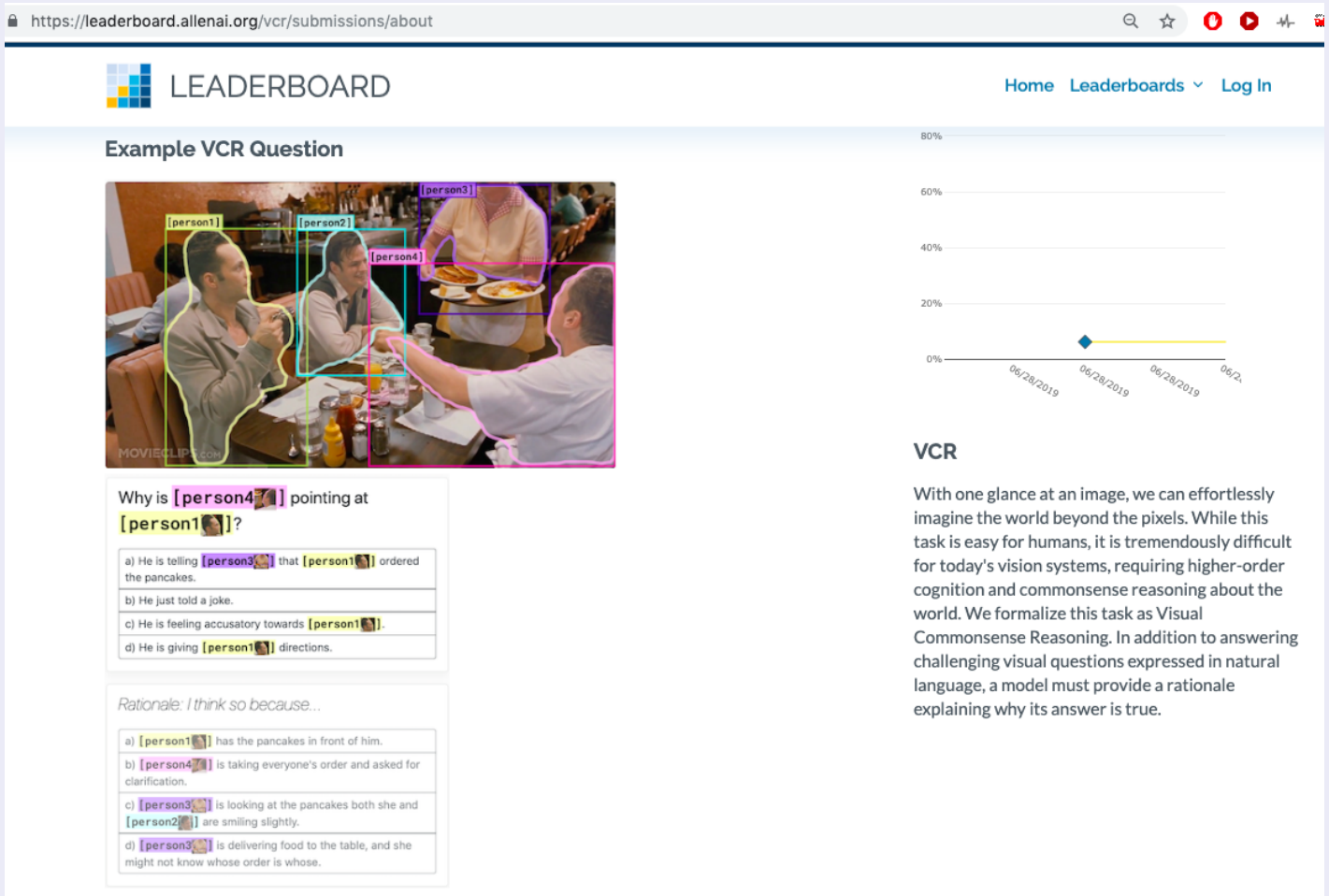
Actions in AI-literature



social qa.png

Actions in AI-literature

visual common sense.png



Pre-AI

Action, Change and Evolution: importance to KR & R

- Historical importance
- Applicability to various domains
- Various knowledge representation aspects
- Various kinds of reasoning

Heracleitos/Herakleitos/Heraclitus of Ephesus (c. 500 BC)

as interpreted by Plato in Cratylus

"No man ever steps in the same river twice, for it is not the same river and he is not the same man."

$\Pi\alpha\nu\tau\alpha\ \rho\epsilon i\ \kappa\alpha i\ o\upsilon\delta\epsilon\nu\ \mu\epsilon\nu\epsilon i$

Panta rei kai ouden menei

All things are in motion and nothing at rest.

alternative

Alternate interpretation of what Heraclitus said

. . . different waters flow in rivers staying the same.

In other words, though the waters are always changing, the rivers stay the same. Indeed, it must be precisely because the waters are always changing that there are rivers at all, rather than lakes or ponds.

The message is that rivers can stay the same over time even though, or indeed because, the waters change. The point, then, is not that everything is changing, but that the fact that some things change makes possible the continued existence of other things.

Free will and choosing ones destiny

366 *THE REPUBLIC OF PLATO.* [BOOK X.

those of the present, Atropos those of the future. Clotho with her right hand takes hold of the outermost rim of the distaff, and twirls it altogether, at intervals; and Atropos with her left hand twirls the inner circles in like manner; while Lachesis takes hold of each in turn with either hand. Now the souls, immediately on their arrival, were required to go to Lachesis. An interpreter first of all marshalled them in order, and then having taken from the lap of Lachesis a number of lots and plans of life, mounted a high pulpit, and spoke as follows: 'Thus saith the maiden Lachesis, the daughter of Necessity. Ye short-lived souls, a new generation of men shall here begin the cycle of its mortal existence. Your destiny shall not be allotted to you, but you shall choose it for yourselves. Let him who draws the first lot be the first to choose a life, which shall be his irrevocably. Virtue owns no master: he who honours her shall have more of her, and he who slights her, less. The responsibility lies with the chooser. Heaven is guiltless.' Having said this, he threw the lots down upon the crowd; and each spirit took up the one which fell by his side, except Er himself, who was forbidden to do so.
618 Each, as he took up his lot, saw what number he had drawn. This done, the plans of life, which far outnumbered the souls that were present, were laid before them on

Where does that line of thought lead us?

- Change is ubiquitous
- But one can shape the change in a desired way
- Some emerging KR issues
  - How to specify change
  - How to specify our desires/goals regarding the change
  - How to construct/verify ways to control the change
  - How to talk about, understand and reason about actions and change

"Action and Change" is encountered often in Computing as well as other fields

- Robots and Agents
- Updates to a database
  Becomes more interesting when updates trigger active rules
- Understanding natural language; interacting in natural langauge
- Distributed Systems
- Computer programs
- Modeling cell behavior
  Ligand coming in contact with a receptor
- Construction Engineering
- ...

Various Kinds of Reasoning

- Prediction
- Plan verification; control verification
- Narratives
- Counterfactuals
- Causal reasoning
- Planning; control generation
- Explanation
- Diagnosis
- Hypothesis generation

Initial Key Issue: Frame Problem

- Motivation: How to specify transition between states of the world due to actions?
  A state transition table would be too space consuming!
- Assume by default that properties of the world normally do not change and specify the exceptions of what changes.
  - How to precisely state the above?
  - Many finer issues!
  - To be elaborate upon as we proceed further.

Origin of the AI "frame" problem

- Leibniz, c.1679
  "everything is presumed to remain in the state in which it is"
- Newton, 1687 (Philosophiae Naturalis Principia Mathematica)
  An object will remain at rest, or continue to move at a constant velocity, unless a resultant force acts on it.

[??]

AXIOMATA
SIVE
LEGES MOTUS

Lex. I.

Corpus omne perſeverare in ſtatu ſuo quieſcendi vel movendi uniformiter in directum, niſi quatenus a viribus impreſſis cogitur ſtatum illum mutare.

Projectilia perſeverant in motibus ſuis niſi quatenus a reſiſtentia aeris retardantur & vi gravitatis impelluntur deorſum. Trochus, cujus partes cohærendo perpetuo retrahunt ſeſe a motibus rectilineis, non ceſſat rotari niſi quatenus ab aere retardatur. Majora autem Planetarum & Cometarum corpora motus ſuos & progreſſivos & circulares in ſpatiis minus reſiſtentibus factos conſervant diutius.

Lex. II.

Mutationem motus proportionalem eſſe vi motrici impreſſæ, & fieri ſecundum lineam rectam qua vis illa imprimitur.

Si vis aliqua motum quemvis generet, dupla duplum, tripla triplum generabit, ſive ſimul & ſemel, ſive gradatim & ſucceſſive impreſſa fuerit. Et hic motus quoniam in eandem ſemper plagam cum vi generatrice determinatur, ſi corpus antea movebatur, motui ejus vel conſpiranti additur, vel contrario ſubducitur, vel obliquo obliquè adjicitur, & cum eo ſecundum utriuſq; determinationem componitur.

Lex. III.

Early work in AI on action and change

- 1959 McCarthy (Programs with common sense)
- 1969 McCarthy and Hayes 1969 (Some philosophical problems from the standpoint of AI) — origin of the "frame problem" in AI
- 1971 Raphael (The frame problem in problem-solving systems) — Defines the frame problem nicely
- 1972 Sandewall (An approach to the frame problem)
- 1972 Hewitt (PLANNER)
- 1973 Hayes (The Frame problem and related problems in AI)
- 1977 Hayes (The logic of frames)
- 1978 Reiter (On reasoning by default)

Quotes from McCarthy & Hayes 1969

- In the last section of part 3, in proving that one person could get into conversation with another, we were obliged to add the hypothesis that if a person has a telephone he still has it after looking up a number in the telephone book. If we had a number of actions to be performed in sequence we would have quite a number of conditions to write down that certain actions do not change the values of certain fluents. In fact with n actions and m fluents we might have to write down mn such conditions.
- We see two ways out of this difficulty. The rest is to introduce the notion of frame, like the state vector in McCarthy (1962). A number of fluents are declared as attached to the frame and the effect of an action is described by telling which fluents are changed, all others being presumed unchanged.

In summary . . .

Action and Change is an important topic in AI

- Its historical basis goes back to pre Plato and Aristotle days
- In AI it goes back to the founding days of AI
- It has a wide applicability
- It involves various kind of KR aspects
- It involves various kinds of reasoning
- It is crucial in understanding and interacting in natural language
- Frame problem in AI identified in AI's early days and has basis in Leibniz's and Newton's work

The Yale Shooting Problem: Hanks & McDermott (AAAI 1986)

- Nonmonotonic formal systems have been proposed as an extension to classical first-order logic that will capture the process of human "default reasoning" or "plausible inference" through their inference mechanisms, just as modus ponens provides a model for deductive reasoning.
- We provide axioms for a simple problem in temporal reasoning which has long been identified as a case of default reasoning, thus presumably amenable to representation in nonmonotonic logic. Upon examining the resulting nonmonotonic theories, however, we find that the inferences permitted by the logics are not those we had intended when we wrote the axioms, and in fact are much weaker. This problem is shown to be independent of the logic used; nor does it depend on any particular temporal representation.
- Upon analyzing the failure we find that the nonmonotonic logics we considered are inherently incapable of representing this kind of default reasoning.

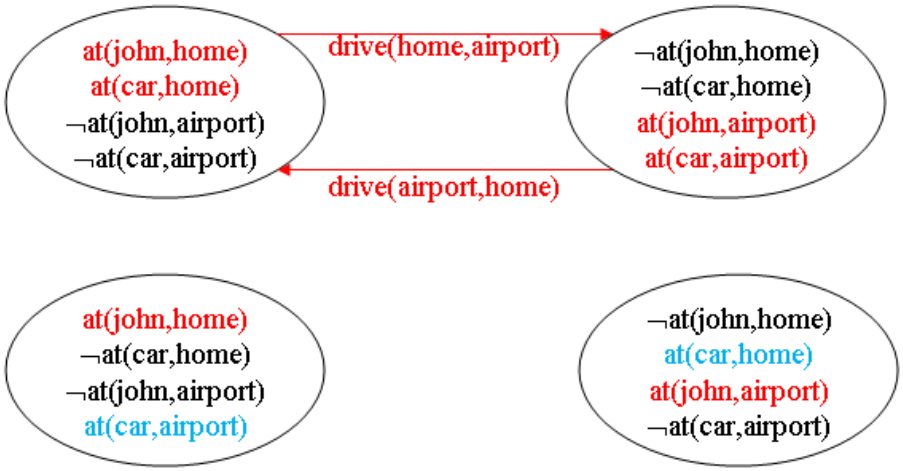
From examples to formal approaches - action languages

- Three schools: Reiter, Sandewall, $\mathcal{A}$ and its follow-ups
- $\mathcal{A}$ Gelfond and Lifschitz (1993): Proposes a solution to the frame problem and assumes complete information.
- Several other languages have been developed to address the ramification problem (or the indirect effects of actions) and consider other aspects of actions:
  - $\mathcal{B}$ (or $\mathcal{AL}$): static causal laws, addressing the ramification problem [Kartha and Lifschitz (1994); Gelfond and Lifschitz (1998); Baral and Gelfond (2000)]
  - $\mathcal{C}$, $\mathcal{C}+$, $\mathcal{BC}$, and $\mathcal{BC}+$: default and dependent fluents [Gelfond and Lifschitz (1998); Giunchiglia and Lifschitz (95); Giunchiglia et al. (1997); Lee and Lifschitz (2003); Lee et al. (2013); Babb and Lee (2015)]
  - Actions with durations, delayed effects [Baral et al. (2002a)]
  - Probability [Baral et al. (2002b); Lee and Wang (2018)]
  - Observations, history [Baral et al. (1997, 2000)]
  - Sensing actions [Son and Baral (2001)]

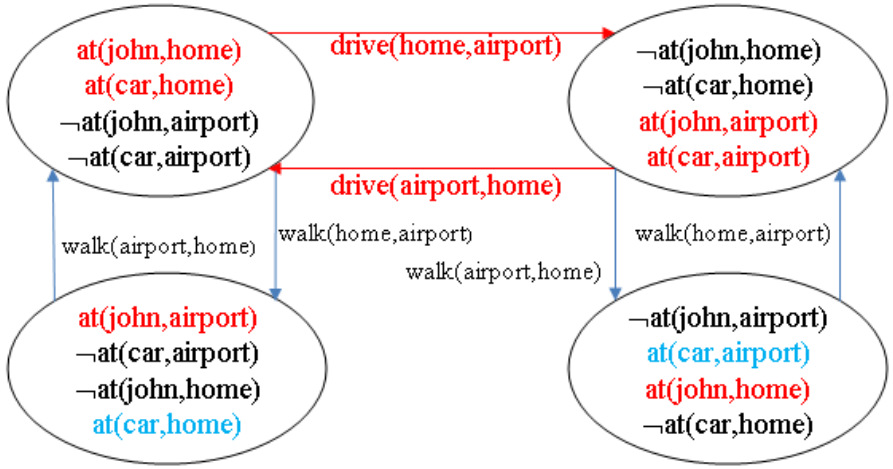
1. Overview: a brief history of action languages **(20 mnt - Chitta)**
2. Action languages in single agent environments **(70 mnt - Son)**
   - The action language $\mathcal{A}$, state, and transition function
   - $\mathcal{AL}$: $\mathcal{A}+$ static causal laws, non-deterministic and sensing actions
   - GOLOG
   - Action languages: related approaches and planning
3. Action languages and causality **(30 mnt - Chitta)**
   - Pearl's do-calculus
4. Action languages in multi-agent environments **(60 mnt – Son)**
   - mA*, Kripke structure, update models, and transition function
   - mA* in epistemic planning
5. Action languages in commonsense reasoning **(18 mnt - Chitta)**
   - Commonsense reasoning and actions
   - Acquiring knowledge about actions
6. Open challenges, conclusion, and discussion **(12 mnt - Chitta)**

Overarching Approach

Dynamic systems as state transitions systems



Going to the Airport



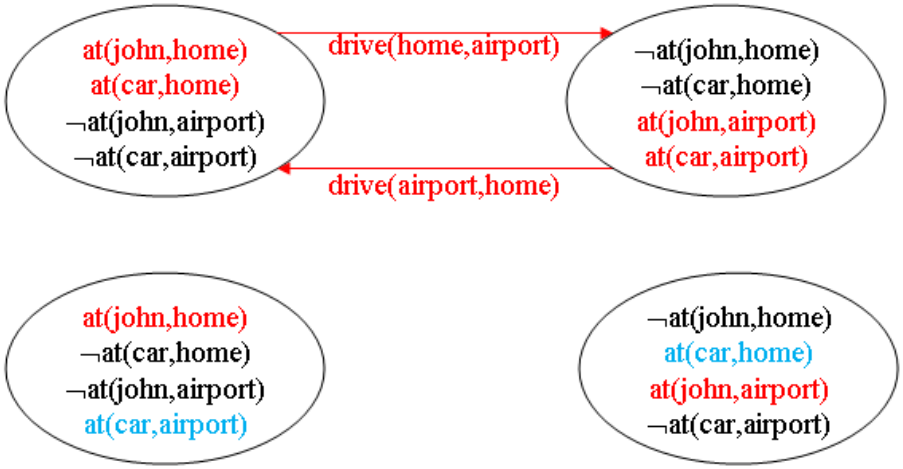
Adding the action $walk(X, Y)$

Two important notions:

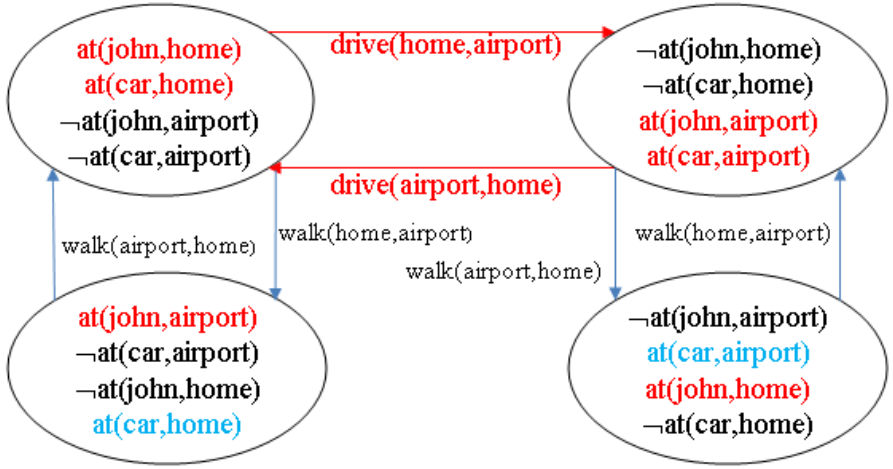
- State?
- Transition?

Overarching Approach

Dynamic systems as state transitions systems



Going to the Airport



Adding the action $walk(X, Y)$

Two important notions: lead to the following questions:

- State - what should be the state?
- Transition - how to define transitions between states?

These are the two questions for any action language

Basic Ontologies (Action Languages, Gelfond and Lifschitz (1993))

Gelfond & Lifschitz. Representing actions in extended logic programs. Journal of Logic Programming, 1993.

- Fluents: property of the world whose value could be changed by actions (e.g., $at(john, home)$)
- Actions: change the state of the world, whose executions create transitions between states of the world (e.g., $drive(home, airport)$)
- Fluent literal: a fluent or its negation (a fluent preceding by $\neg$)
  E.g. $at(john, home)$, $\neg at(john, home)$
- State: two commonly used definitions
  - a *set of fluents* ($s \subseteq F$: whatever is in s is true; otherwise, it is false) or
  - a *complete* and *consistent* set of fluent literals, i.e., s is a state if for every fluent f
    - ⋆ either f or $\neg f$ belongs to s; and
    - ⋆ $\{f, \neg f\} \not\subseteq s$.

Notion: a literal l is true in a state s

Language $\mathcal{A}$ — Syntax

In $\mathcal{A}$, an action theory is defined over two disjoint sets, a set of fluents F and a set of actions A, and is a set of statements of the form

$$a \textbf{ causes } f \textbf{ if } p_1, \ldots, p_n \quad (1)$$

$$a \textbf{ executable\_if } p_1 \ldots, p_n \quad (2)$$

$$\textbf{initially } f \quad (3)$$

where f and p_i's are fluent literals (a *fluent literal* is either a fluent g or its negation $\neg g$) and a is an action.

Dynamic Law

A statement of the form (1): a **causes** f **if** $p_1, \ldots, p_n$
is called a dynamic law. It represents the (conditional) effect of action a.
It says that if a is executed and $p_1, \ldots, p_n$ are true then f becomes true.

Dynamic Law: Examples

- Stacking a block X on top of block Y causes X to be on Y, X is clear, Y is no longer clear, and the agent does not hold anything can be expressed by

Dynamic Law

A statement of the form (1): a **causes** f **if** $p_1, \ldots, p_n$
is called a dynamic law. It represents the (conditional) effect of action a.
It says that if a is executed and $p_1, \ldots, p_n$ are true then f becomes true.

Dynamic Law: Examples

- Stacking a block X on top of block Y causes X to be on Y, X is clear, Y is no longer clear, and the agent does not hold anything can be expressed by
  $stack(X, Y)$ **causes** $on(X, Y)$
  $stack(X, Y)$ **causes** $clear(X)$
  $stack(X, Y)$ **causes** $\neg clear(Y)$
  $stack(X, Y)$ **causes** $handEmpty$
  Shorthand formalization:
  $stack(X, Y)$ **causes** $on(X, Y), clear(X), \neg clear(Y), handEmpty$
- Shooting causes the turkey to be dead and the gun becomes unloaded if the gun is loaded can be expressed by

Dynamic Law

A statement of the form (1): a **causes** f **if** $p_1, \ldots, p_n$
is called a dynamic law. It represents the (conditional) effect of action a.
It says that if a is executed and $p_1, \ldots, p_n$ are true then f becomes true.

Dynamic Law: Examples

- Stacking a block X on top of block Y causes X to be on Y, X is clear, Y is no longer clear, and the agent does not hold anything can be expressed by
  $stack(X, Y)$ **causes** $on(X, Y)$
  $stack(X, Y)$ **causes** $clear(X)$
  $stack(X, Y)$ **causes** $\neg clear(Y)$
  $stack(X, Y)$ **causes** $handEmpty$
  Shorthand formalization:
  $stack(X, Y)$ **causes** $on(X, Y), clear(X), \neg clear(Y), handEmpty$
- Shooting causes the turkey to be dead and the gun becomes unloaded if the gun is loaded can be expressed by
  $shoot$ **causes** $dead$ **if** $loaded$ and
  $shoot$ **causes** $\neg loaded$ **if** $loaded$

Executability Condition (Preconditions)

A statement of the form (2):

$$a \textbf{ executable\_if } p_1 \ldots, p_n$$

is a executability condition statement. It states that a can be executed only if $p_1, \ldots, p_n$ are true.

Examples

- A gun can be loaded only when it is not loaded

Executability Condition (Preconditions)

A statement of the form (2):

$$a \textbf{ executable\_if } p_1 \ldots, p_n$$

is a executability condition statement. It states that a can be executed only if $p_1, \ldots, p_n$ are true.

Examples

- A gun can be loaded only when it is not loaded
  load **executable_if** $\neg$*loaded*
- One can pick up a block X only if one's hand is empty, X is clear, and X is on the table

Executability Condition (Preconditions)

A statement of the form (2):

$$a \textbf{ executable\_if } p_1 \ldots, p_n$$

is a executability condition statement. It states that a can be executed only if $p_1, \ldots, p_n$ are true.

Examples

- A gun can be loaded only when it is not loaded
  load **executable_if** $\neg$*loaded*
- One can pick up a block X only if one's hand is empty, X is clear, and X is on the table
  $pickup(X)$ **executable_if** $onTable(X), clear(X), handEmpty$

Initial State

A statement of the form (3):

initially f

is a initial state statement. It states that f is true in the initial state.

Examples

- Initially, the gun is loaded:

Initial State

A statement of the form (3):

initially f

is a initial state statement. It states that f is true in the initial state.

Examples

- Initially, the gun is loaded:
  initially *loaded*
- Initially, a is on the table, c is on the table, b is on c.

Initial State

A statement of the form (3):

$$\textbf{initially} \quad f$$

is a initial state statement. It states that f is true in the initial state.

Examples

- Initially, the gun is loaded:
  initially *loaded*
- Initially, a is on the table, c is on the table, b is on c.
  initially $onTable(a)$
  initially $onTable(c)$
  initially $on(b, c)$

Action Theory

An action theory is given by a pair (D, I) where D consists of statements of the form (1)-(2) and I consists of propositions of the form (3).

Yale Shooting Problem

Represented as an action theory (D_y, I_y)

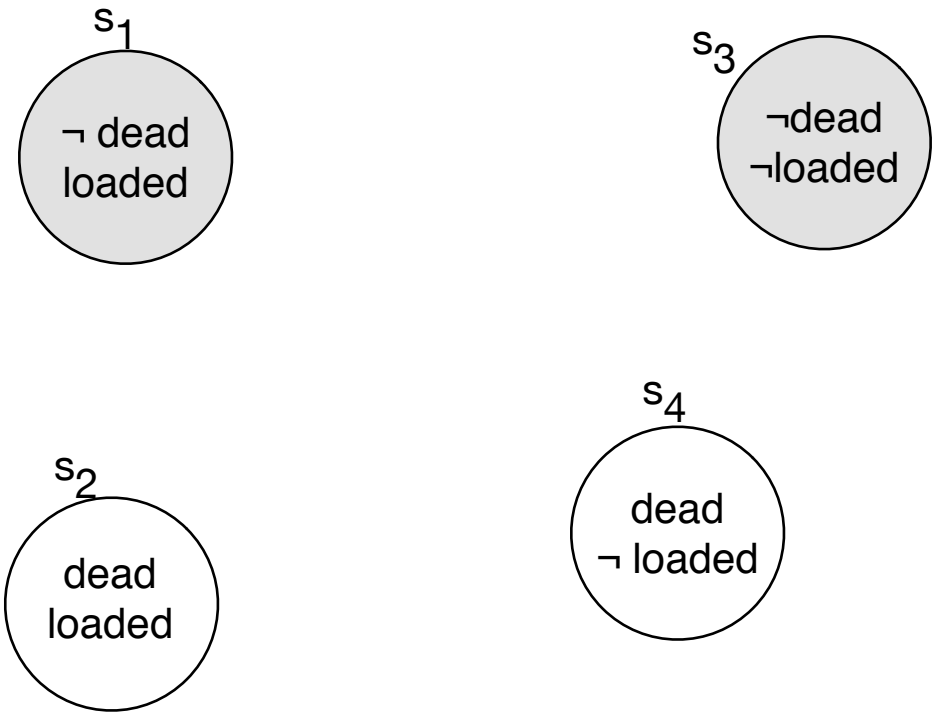
$$I_y = \{ \textbf{initially} \ \neg dead, \ \textbf{initially} \ loaded \}$$

and

$$D_y = \left\{ \begin{array}{l} shoot \ \textbf{causes} \ dead \ \textbf{if} \ loaded \\ shoot \ \textbf{causes} \ \neg loaded \ \textbf{if} \ loaded \\ load \ \textbf{causes} \ loaded \\ shoot \ \textbf{executable\_if} \ true \\ load \ \textbf{executable\_if} \ \neg loaded \end{array} \right\}$$

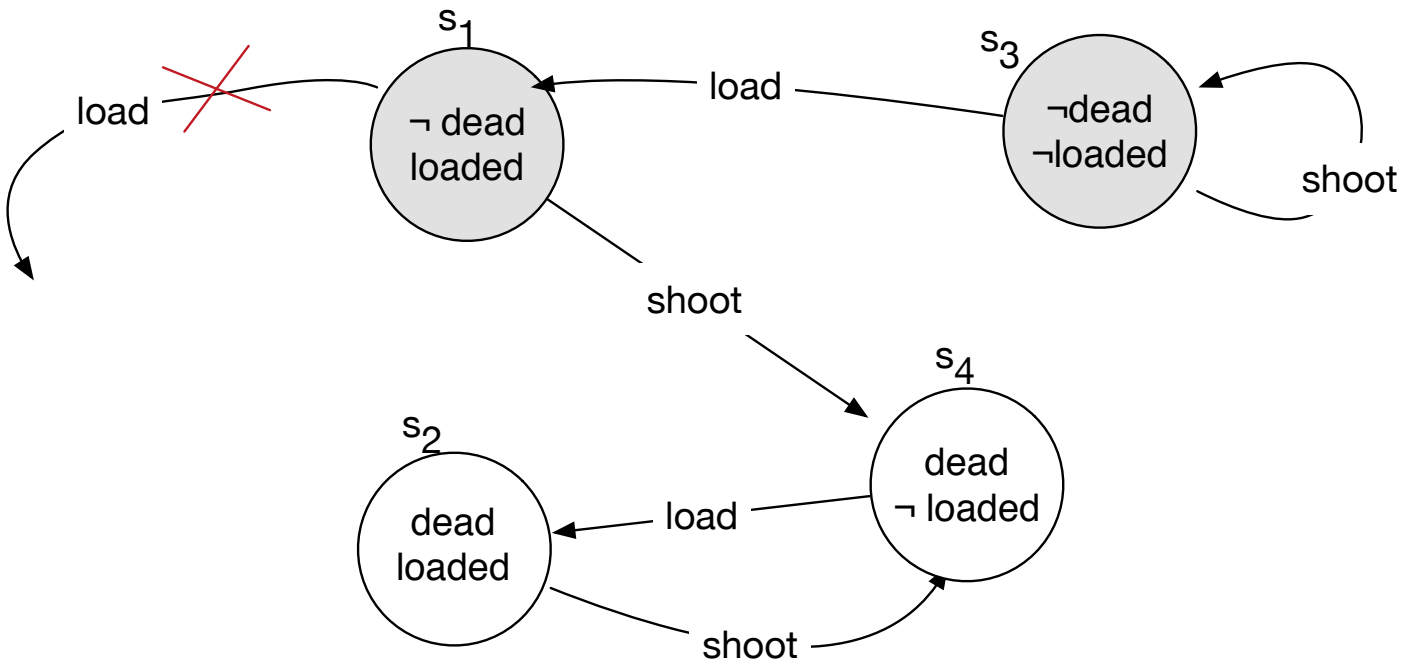
Language $\mathcal{A}$ - Semantics: Intuition

The set of fluents define the states in an action theory.



Language $\mathcal{A}$ - Semantics: Intuition

The set of fluents define the states in an action theory.
The set of dynamic laws specify the transitions between states in the domain



States and Transitions | What is a state?

Let D be a domain with a set of fluents F.

A state s of F is a complete and consistent set of literals constructed from F.

- complete: $\forall f \in F.[f \in s \vee \neg f \in s]$
- consistent: $\forall f \in F.[\neg(f \in s \wedge \neg f \in s)]$

Following are some the states in the Yale shooting domain:

$s_1 = \{\neg dead, loaded\}$
$s_2 = \{dead, loaded\}$
$s_3 = \{\neg dead, \neg loaded\}$
$s_4 = \{dead, \neg loaded\}$

A fluent f is said to be true (resp. false) in a state s iff $f \in s$ (resp. $\neg f \in s$).

States and Transitions | What is a transition?

- An action a is executable in a state s if there exists a **executable_if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ are true in s. Clearly, if a **executable_if** *true* belongs to D, then a is executable in every state of D.
- The set of effects of an action a in a state s is the set

$$e(a, s) = \{f \mid a \textbf{ causes } f \textbf{ if } \ p_1, \ldots, p_n \in D, p_i \text{ is true in } s\}.$$

Define

$$\overline{e(a, s)} = \{\neg f \mid f \in F \cap e(a, s)\} \cup \{f \mid f \in F, \neg f \in e(a, s)\}$$

- For a domain D, $\Phi(a, s)$, the state resulting from executing a in s, is defined as follows.
  1. If a is executable in s, then
  $$\Phi(a, s) = s \setminus \overline{e(a, s)} \cup e(a, s)$$
  2. If a is not executable in s, then $\Phi(a, s) = \textit{undefined}$.

States and Transitions: Examples

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, loaded\}$
$s_2 = \{dead, loaded\}$
$s_3 = \{\neg dead, \neg loaded\}$
$s_4 = \{dead, \neg loaded\}$

$\Phi(shoot, s_1) =$
$\Phi(load, s_1) =$
$\Phi(shoot, s_2) =$
$\Phi(load, s_2) =$
$\Phi(shoot, s_3) =$
$\Phi(load, s_3) =$
$\Phi(shoot, s_4) =$
$\Phi(load, s_4) =$

States and Transitions: Examples

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$$s_1 = \{\neg dead, loaded\}$$
$$s_2 = \{dead, loaded\}$$
$$s_3 = \{\neg dead, \neg loaded\}$$
$$s_4 = \{dead, \neg loaded\}$$

$\Phi(shoot, s_1) = s_1 \setminus \overline{\{dead, \neg loaded\}} \cup \{dead, \neg loaded\}$
$\{dead, \neg loaded\}$

$\Phi(load, s_1) =$

$\Phi(shoot, s_2) =$

$\Phi(load, s_2) =$

$\Phi(shoot, s_3) =$

$\Phi(load, s_3) =$

$\Phi(shoot, s_4) =$

$\Phi(load, s_4) =$

States and Transitions: Examples

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$$s_1 = \{\neg dead, loaded\}$$
$$s_2 = \{dead, loaded\}$$
$$s_3 = \{\neg dead, \neg loaded\}$$
$$s_4 = \{dead, \neg loaded\}$$

$\Phi(shoot, s_1) = s_1 \setminus \overline{\{dead, \neg loaded\}} \cup \{dead, \neg loaded\}$
$\{dead, \neg loaded\}$

$\Phi(load, s_1) = undefined$ (load cannot execute load in s_1)

$\Phi(shoot, s_2) =$

$\Phi(load, s_2) =$

$\Phi(shoot, s_3) =$

$\Phi(load, s_3) =$

$\Phi(shoot, s_4) =$

$\Phi(load, s_4) =$

States and Transitions: Examples

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$$s_1 = \{\neg dead, loaded\}$$
$$s_2 = \{dead, loaded\}$$
$$s_3 = \{\neg dead, \neg loaded\}$$
$$s_4 = \{dead, \neg loaded\}$$

$\Phi(shoot, s_1) = s_1 \setminus \overline{\{dead, \neg loaded\}} \cup \{dead, \neg loaded\}$
$\{dead, \neg loaded\}$

$\Phi(load, s_1) = undefined$ (load cannot execute load in s_1)

$\Phi(shoot, s_2) = \{dead, \neg loaded\}$

$\Phi(load, s_2) =$

$\Phi(shoot, s_3) =$

$\Phi(load, s_3) =$

$\Phi(shoot, s_4) =$

$\Phi(load, s_4) =$

States and Transitions: Examples

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$$s_1 = \{\neg dead, loaded\}$$
$$s_2 = \{dead, loaded\}$$
$$s_3 = \{\neg dead, \neg loaded\}$$
$$s_4 = \{dead, \neg loaded\}$$

$$\Phi(shoot, s_1) = s_1 \setminus \overline{\{dead, \neg loaded\}} \cup \{dead, \neg loaded\}$$
$$\{dead, \neg loaded\}$$

$\Phi(load, s_1) = undefined$ (load cannot execute load in s_1)

$\Phi(shoot, s_2) = \{dead, \neg loaded\}$

$\Phi(load, s_2) = undefined$

$\Phi(shoot, s_3) = \{\neg dead, \neg loaded\}$

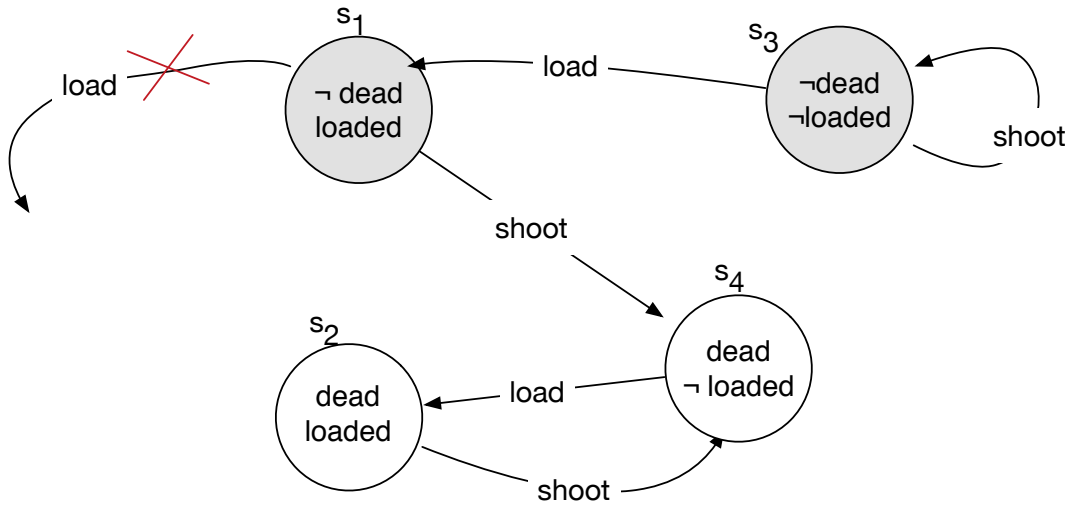
$\Phi(load, s_3) = \{\neg dead, loaded\}$

$\Phi(shoot, s_4) = \{\neg dead, \neg loaded\}$

$\Phi(load, s_4) = \{\neg dead, loaded\}$

Transition System of Yale Shooting

$$s_1 = \{\neg dead, loaded\}$$
$$s_2 = \{dead, loaded\}$$
$$s_3 = \{\neg dead, \neg loaded\}$$
$$s_4 = \{dead, \neg loaded\}$$



Intuition for $\Phi(a, s)$?

Intuition

When an action is executed, the following can happen

- the action directly changes some fluents;
- other fluents stay unchanged;

Intuition for $\Phi(a, s)$?

Intuition

When an action is executed, the following can happen

- the action directly changes some fluents; this is what in $e(a, s)$!
- other fluents stay unchanged;

Intuition for $\Phi(a, s)$?

Intuition

When an action is executed, the following can happen

- the action directly changes some fluents; this is what in $e(a, s)$!
- other fluents stay unchanged; this is what in $s \setminus \overline{e(a, s)}$! [**frame problem**]

Intuition for $\Phi(a, s)$?

Intuition

When an action is executed, the following can happen

- the action directly changes some fluents; this is what in $e(a, s)$!
- other fluents stay unchanged; this is what in $s \setminus \overline{e(a, s)}$! **[frame problem]**

Given a, s, and D, assume that a is executable in s
$e(a, s) = \{f \mid a \textbf{ causes } f \textbf{ if } p_1, \ldots, p_n \in D, p_i \text{ is true in } s\}$
$\Phi(a, s) = s \setminus \overline{e(a, s)} \cup e(a, s)$

Reasoning about Plan: the entailment ($\models$) relation

$\Phi(a, s)$: the result of executing the action a in a state s.
Consider a sequence of actions (or plan) $\alpha = [a_1; \ldots; a_n]$
What are true/false after the execution of α from the initial state?

$(D, I) \models l$ **after** α?

Define $\widehat{\Phi}(\alpha, s)$.

- $\widehat{\Phi}([], s) = s$
- if a is executable in s then $\widehat{\Phi}([a, \beta], s) = \widehat{\Phi}(\beta, \Phi(a, s))$; otherwise, $\widehat{\Phi}([a, \beta], s) = \textit{undefined}$

Let s_0 be the initial state:

$$(D, I) \models l \textbf{ after } \alpha$$

iff $\widehat{\Phi}(\alpha, s_0)$ is defined and l is true in $\widehat{\Phi}(\alpha, s_0)$.

$\models$: Example

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases} \qquad I_y = \begin{cases} \textbf{initially } \neg\textit{loaded} \\ \textbf{initially } \neg\textit{dead} \end{cases}$$

Initial state: $s_0 = \{\neg dead, \neg loaded\}$

- $(D_y, I_y) \models loaded$ **after** $[load]$ because
  $\Phi(load, s_0) = \{\neg dead, loaded\}$.
- $(D_y, I_y) \models dead$ **after** $[load, shoot]$ because
  $\widehat{\Phi}([load, shoot], s_0) = \Phi(shoot, \{\neg dead, loaded\}) = \{dead, \neg loaded\}$.

Static Causal Law

Fluents are related to each others.
Sometime, a change of a fluent's value causes other fluents to change.
This is often referred to as **indirect effects (of actions)** or the **ramification problem** in RAC.

Examples

- Dead turkeys cannot walk
  $\neg walking$ **if** $dead$
- One block cannot be on top of two different blocks
  $false$ **if** $on(X, Y), on(X, Z), Y \neq Z, X \neq Y$
- A block is on top of another block cannot be on the table
  $\neg onTable(X)$ **if** $on(X, Y), X \neq Y$

Static Causal Law

A statement of the form

$$f \textbf{ if } p_1, \ldots, p_n \tag{4}$$

is a **static causal law** which represents the relationship between fluents. It is a constraint stating that whenever $p_1, \ldots, p_n$ are true then f must be true.

Examples

- Dead turkeys cannot walk
  $\neg walking$ **if** $dead$
- One block cannot be on top of two different blocks
  $false$ **if** $on(X, Y), on(X, Z), Y \neq Z, X \neq Y$
- A block is on top of another block cannot be on the table
  $\neg onTable(X)$ **if** $on(X, Y), X \neq Y$

$\mathcal{AL}$ Action Theory

An action theory is given by a pair (D, I) where D consists of statements of the form (1)-(2) and (4) and I consists of statements of the form (3).

Yale Shooting Problem with Static Causal Laws

Represented as an action theory (D_y^w, I_y^w)

$$I_y^w = \{ \textbf{initially} \ \neg dead, \ \textbf{initially} \ walking, \ \textbf{initially} \ loaded \}$$

and

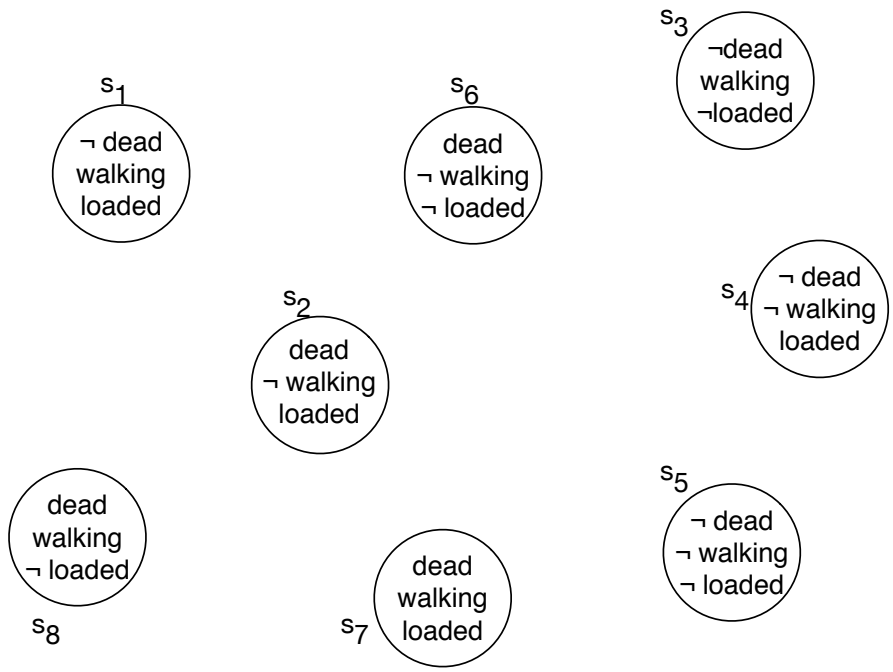
$$D_y^w = \left\{ \begin{array}{l} shoot \ \textbf{causes} \ dead \ \textbf{if} \ loaded \\ shoot \ \textbf{causes} \ \neg loaded \ \textbf{if} \ loaded \\ load \ \textbf{causes} \ loaded \\ \neg walking \ \textbf{if} \ dead \\ shoot \ \textbf{executable\_if} \ true \\ load \ \textbf{executable\_if} \ \neg loaded \end{array} \right\}$$

States and Transitions in $\mathcal{AL}$ What is a state?

Can state be the same as in $\mathcal{A}$ (a complete and consistent set of literals)?
The presence of static causal laws removes some potential states

Potential states:

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$
$s_4 = \{\neg dead, \neg walking, loaded\}$
$s_5 = \{\neg dead, \neg walking, \neg loaded\}$
$s_6 = \{dead, \neg walking, \neg loaded\}$
$s_7 = \{dead, walking, loaded\}$
$s_8 = \{dead, walking, \neg loaded\}$

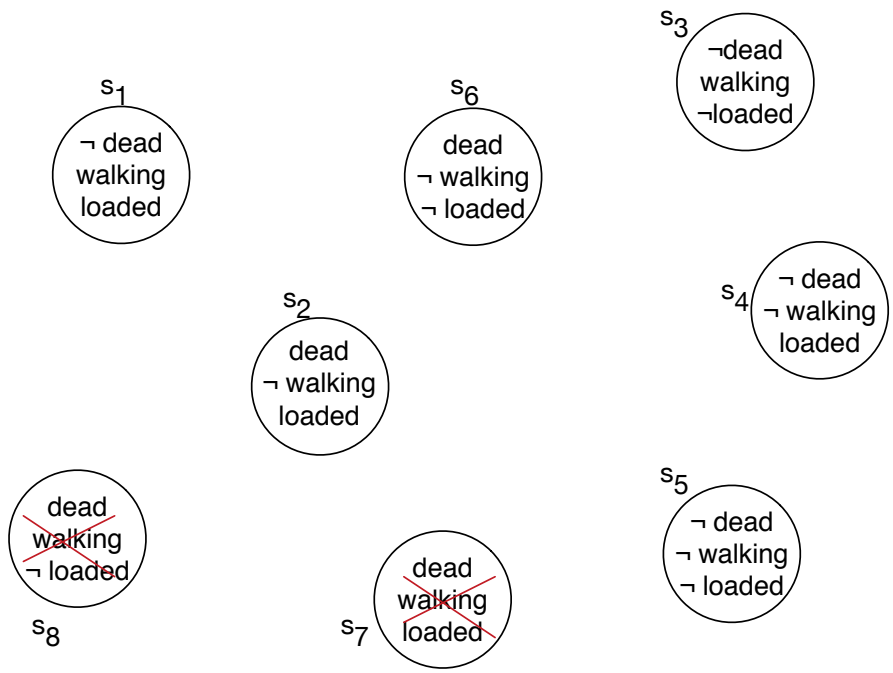


States and Transitions in $\mathcal{AL}$ What is a state?

Can state be the same as in $\mathcal{A}$ (a complete and consistent set of literals)?
The presence of static causal laws removes some potential states

Potential states:

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$
$s_4 = \{\neg dead, \neg walking, loaded\}$
$s_5 = \{\neg dead, \neg walking, \neg loaded\}$
$s_6 = \{dead, \neg walking, \neg loaded\}$
$s_7 = \{dead, walking, loaded\}$
$s_8 = \{dead, walking, \neg loaded\}$



Need a way to eliminate unrealistic states!

States and Transitions in $\mathcal{AL}$ What is a state? — $Cn(X)$

Given a set of literals X, $Cn(X)$ is the minimal set of literals that

- contains X and
- satisfies all static laws in D.

Yale Shooting Domain

Has one static law: $\neg walking$ **if** $dead$

$x_1 = \{\neg dead\}$
$x_2 = \{dead, loaded\}$
$x_3 = \{walking, \neg loaded\}$
$x_4 = \{dead, walking, \neg loaded\}$

States and Transitions in $\mathcal{AL}$ What is a state? — $Cn(X)$

Given a set of literals X, $Cn(X)$ is the minimal set of literals that

- contains X and
- satisfies all static laws in D.

Yale Shooting Domain

Has one static law: $\neg walking$ **if** $dead$

$x_1 = \{\neg dead\}$ $Cn(x_1) = \{\neg dead\}$

$x_2 = \{dead, loaded\}$

$x_3 = \{walking, \neg loaded\}$

$x_4 = \{dead, walking, \neg loaded\}$

States and Transitions in $\mathcal{AL}$ What is a state? — $Cn(X)$

Given a set of literals X, $Cn(X)$ is the minimal set of literals that

- contains X and
- satisfies all static laws in D.

Yale Shooting Domain

Has one static law: $\neg walking$ **if** $dead$

$x_1 = \{\neg dead\}$ $Cn(x_1) = \{\neg dead\}$

$x_2 = \{dead, loaded\}$ $Cn(x_2) = \{dead, \neg walking, loaded\}$

$x_3 = \{walking, \neg loaded\}$

$x_4 = \{dead, walking, \neg loaded\}$

States and Transitions in $\mathcal{AL}$ What is a state? — $Cn(X)$

Given a set of literals X, $Cn(X)$ is the minimal set of literals that

- contains X and
- satisfies all static laws in D.

Yale Shooting Domain

Has one static law: $\neg walking$ **if** $dead$

$x_1 = \{\neg dead\}$ $Cn(x_1) = \{\neg dead\}$

$x_2 = \{dead, loaded\}$ $Cn(x_2) = \{dead, \neg walking, loaded\}$

$x_3 = \{walking, \neg loaded\}$ $Cn(x_3) = \{walking, \neg loaded\}$

$x_4 = \{dead, walking, \neg loaded\}$

States and Transitions in $\mathcal{AL}$ What is a state? — $Cn(X)$

Given a set of literals X, $Cn(X)$ is the minimal set of literals that

- contains X and
- satisfies all static laws in D.

Yale Shooting Domain

Has one static law: $\neg walking$ **if** $dead$

$x_1 = \{\neg dead\}$ $Cn(x_1) = \{\neg dead\}$

$x_2 = \{dead, loaded\}$ $Cn(x_2) = \{dead, \neg walking, loaded\}$

$x_3 = \{walking, \neg loaded\}$ $Cn(x_3) = \{walking, \neg loaded\}$

$x_4 = \{dead, walking, \neg loaded\}$

$Cn(x_4) = \{dead, walking, \neg walking, \neg loaded\}$

Sometime, $Cn(X)$ is inconsistent!

States and Transitions in $\mathcal{AL}$ What is a state? — $Cn(X)$

Given a set of literals X, $Cn(X)$ is the minimal set of literals that

- contains X and
- satisfies all static laws in D.

Yale Shooting Domain

Has one static law: $\neg walking$ **if** $dead$

$x_1 = \{\neg dead\}$ $Cn(x_1) = \{\neg dead\}$

$x_2 = \{dead, loaded\}$ $Cn(x_2) = \{dead, \neg walking, loaded\}$

$x_3 = \{walking, \neg loaded\}$ $Cn(x_3) = \{walking, \neg loaded\}$

$x_4 = \{dead, walking, \neg loaded\}$

$Cn(x_4) = \{dead, walking, \neg walking, \neg loaded\}$

Sometime, $Cn(X)$ is inconsistent!

States in $\mathcal{AL}$

A complete and consistent set of literals s is a state if $s = Cn(s)$.

States and Transitions in $\mathcal{AL}$ | What is a transition?

The notion of executability of an action a in a state s remains the same: if there exists an executability proposition a **executable_if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ are true in s then a is executable in s.
The set of effects of an action a in a state s is the set

$$e(a, s) = \{f \mid a \textbf{ causes } f \textbf{ if } \ p_1, \ldots, p_n \in D, p_i \text{ is true in } s\}.$$

For a domain description D, $\Phi(a, s)$, the set of states that may be reached by executing a in s, is defined as follows.

1. If a is executable in s, then

$$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e(a, s))\}$$

2. If a is not executable in s, then $\Phi(a, s) = \textit{undefined}$.

Why is $\Phi(a, s)$ defined like this?

Intuition

When an action is executed in s and the result is a state s', the following can happen

- the action directly changes some fluents;
- some fluents stay unchanged;
- some fluents were changed indirectly.

Given a, s, and D, assume that a is executable in s
$e(a, s) = \{f \mid a$ **causes** f **if** $p_1, \ldots, p_n \in D, p_i$ is true in $s\}$.
$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e(a, s))\}$

Why is $\Phi(a, s)$ defined like this?

Intuition

When an action is executed in s and the result is a state s', the following can happen

- the action directly changes some fluents; this is what in $e(a, s)$!
- some fluents stay unchanged;
- some fluents were changed indirectly.

Given a, s, and D, assume that a is executable in s
$e(a, s) = \{f \mid a$ **causes** f **if** $p_1, \ldots, p_n \in D, p_i$ is true in $s\}$.
$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e(a, s))\}$

Why is $\Phi(a, s)$ defined like this?

Intuition

When an action is executed in s and the result is a state s', the following can happen

- the action directly changes some fluents; this is what in $e(a, s)$!
- some fluents stay unchanged; this is what in $s \cap s'$! **[frame problem]**
- some fluents were changed indirectly.

Given a, s, and D, assume that a is executable in s
$e(a, s) = \{f \mid a$ **causes** f **if** $p_1, \ldots, p_n \in D, p_i$ is true in $s\}$.
$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e(a, s))\}$

Why is $\Phi(a, s)$ defined like this?

Intuition

When an action is executed in s and the result is a state s', the following can happen

- the action directly changes some fluents; this is what in $e(a, s)$!
- some fluents stay unchanged; this is what in $s \cap s'$! **[frame problem]**
- some fluents were changed indirectly. this is what in $Cn((s \cap s') \cup e(a, s))$! **[ramification problem]**

Given a, s, and D, assume that a is executable in s
$e(a, s) = \{f \mid a$ **causes** f **if** $p_1, \ldots, p_n \in D, p_i$ is true in $s\}$.
$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e(a, s))\}$

States and Transitions in $\mathcal{AL}$: Examples

$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) =$
$\Phi(load, s_1) =$
$\Phi(shoot, s_2) =$
$\Phi(load, s_2) =$
$\Phi(shoot, s_3) =$
$\Phi(load, s_3) =$

States and Transitions in $\mathcal{AL}$: Examples

$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_1) =$
$\Phi(shoot, s_2) =$
$\Phi(load, s_2) =$
$\Phi(shoot, s_3) =$
$\Phi(load, s_3) =$

States and Transitions in $\mathcal{AL}$: Examples

$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_1) = undefined$ (cannot execute load in s_1)
$\Phi(shoot, s_2) =$
$\Phi(load, s_2) =$
$\Phi(shoot, s_3) =$
$\Phi(load, s_3) =$

States and Transitions in $\mathcal{AL}$: Examples

$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_1) = undefined$ (cannot execute load in s_1)
$\Phi(shoot, s_2) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_2) =$
$\Phi(shoot, s_3) =$
$\Phi(load, s_3) =$

States and Transitions in $\mathcal{AL}$: Examples

$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_1) = undefined$ (cannot execute load in s_1)
$\Phi(shoot, s_2) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_2) = undefined$
$\Phi(shoot, s_3) =$
$\Phi(load, s_3) =$

States and Transitions in $\mathcal{AL}$: Examples

$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_1) = undefined$ (cannot execute load in s_1)
$\Phi(shoot, s_2) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_2) = undefined$
$\Phi(shoot, s_3) = \{\{\neg dead, walking, \neg loaded\}\}$
$\Phi(load, s_3) =$

States and Transitions in $\mathcal{AL}$: Examples

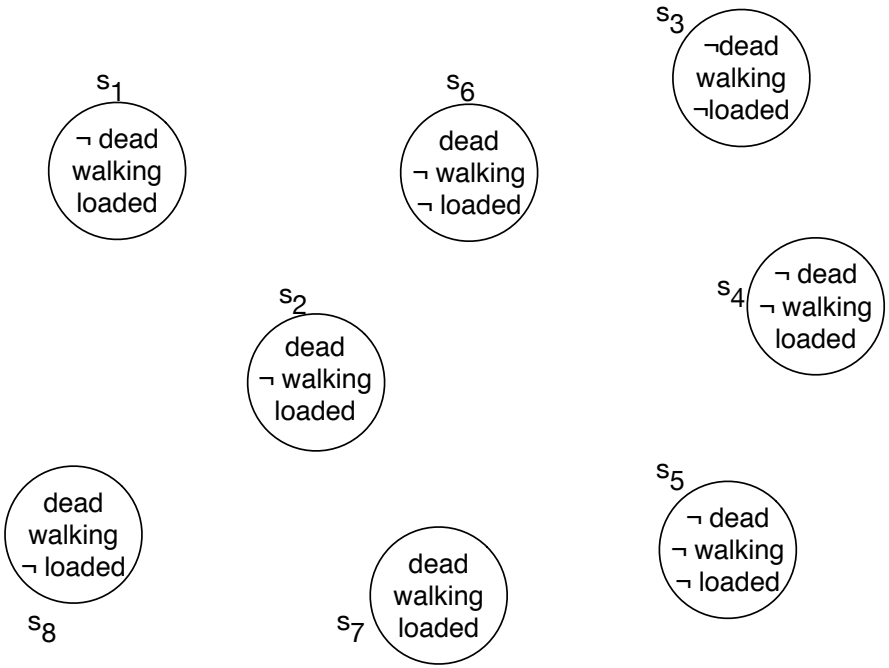
$$D_y^w = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$

$\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_1) = undefined$ (cannot execute load in s_1)
$\Phi(shoot, s_2) = \{\{dead, \neg loaded, \neg walking\}\}$
$\Phi(load, s_2) = undefined$
$\Phi(shoot, s_3) = \{\{\neg dead, walking, \neg loaded\}\}$
$\Phi(load, s_3) = \{\{\neg dead, walking, loaded\}\}$

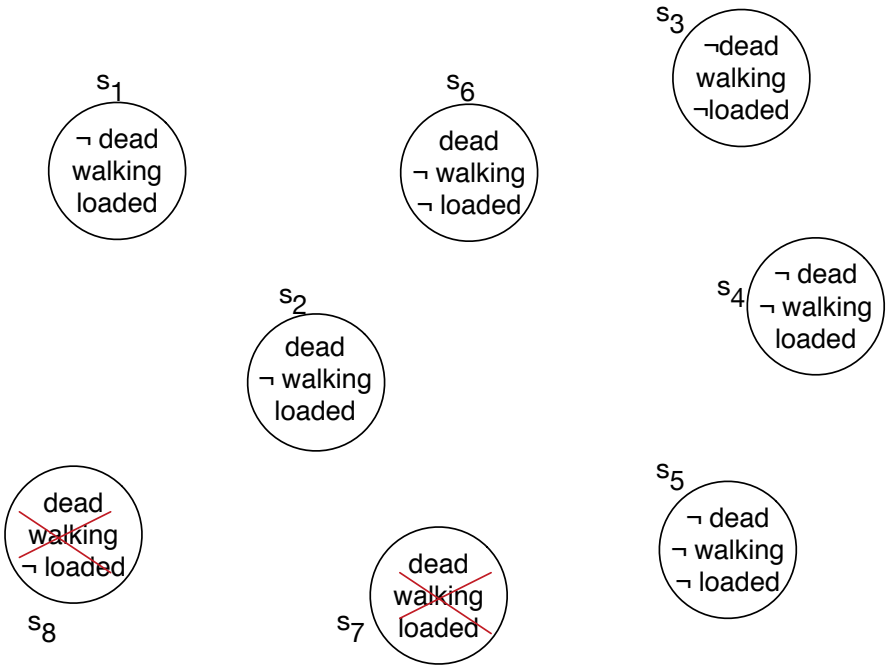
Transition System of Yale Shooting with Causal Laws

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$
$s_4 = \{\neg dead, \neg walking, loaded\}$
$s_5 = \{\neg dead, \neg walking, \neg loaded\}$
$s_6 = \{dead, \neg walking, \neg loaded\}$
$s_7 = \{dead, walking, loaded\}$
$s_8 = \{dead, walking, \neg loaded\}$



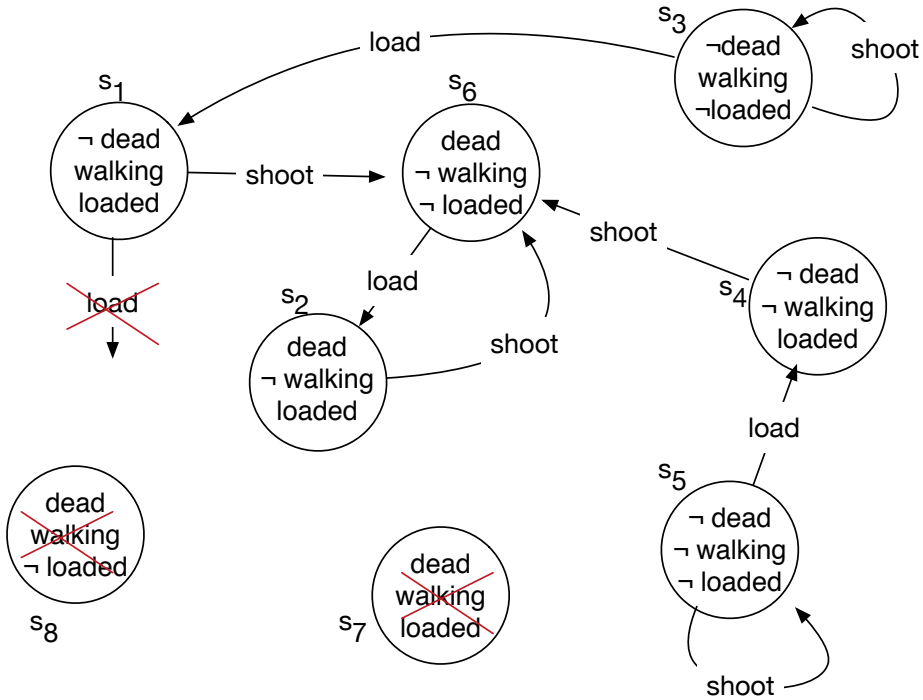
Transition System of Yale Shooting with Causal Laws

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$
$s_4 = \{\neg dead, \neg walking, loaded\}$
$s_5 = \{\neg dead, \neg walking, \neg loaded\}$
$s_6 = \{dead, \neg walking, \neg loaded\}$
$s_7 = \{dead, walking, loaded\}$
$s_8 = \{dead, walking, \neg loaded\}$



Transition System of Yale Shooting with Causal Laws

$s_1 = \{\neg dead, walking, loaded\}$
$s_2 = \{dead, \neg walking, loaded\}$
$s_3 = \{\neg dead, walking, \neg loaded\}$
$s_4 = \{\neg dead, \neg walking, loaded\}$
$s_5 = \{\neg dead, \neg walking, \neg loaded\}$
$s_6 = \{dead, \neg walking, \neg loaded\}$
$s_7 = \{dead, walking, loaded\}$
$s_8 = \{dead, walking, \neg loaded\}$



How To Compute $\Phi(a, s)$?

Given a, s, and D, assume that a is executable in s
$e(a, s) = \{f \mid a$ **causes** f **if** $p_1, \ldots, p_n \in D, p_i$ is true in $s\}$.
$\Phi(a, s) = \{s' \mid Cn(s') = Cn((s \cap s') \cup e(a, s))\}$

Computing $\Phi(a, s)$

1. Compute $e(a, s)$
2. Eliminate from s those that are false in $e(a, s)$
3. Identify maximal sets of atoms X that remain in s and can be joined together with $e(a, s)$ such that $s' = Cn(X \cup e(a, s))$ is a consistent set of literals and is an interpretation.

States and Transitions: Example

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \end{cases} \qquad \begin{array}{l} \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{array}$$

$s_1 = \{\neg dead, walking, loaded\}$. Computer $\Phi(shoot, s_1)$

1. Compute $e(shoot, s_1) =$
2. Eliminate from s_1 those that are false in $e(shoot, s_1)$:
3. Identify maximal sets of atoms X that remain in s_1 and can be joined together with $e(shoot, s_1)$ s.t. $s' = Cn(X \cup e(shoot, s_1))$ is a consistent set of literals and is an interpretation.

States and Transitions: Example

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \end{cases} \qquad \begin{array}{l} \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{array}$$

$s_1 = \{\neg dead, walking, loaded\}$. Computer $\Phi(shoot, s_1)$

1. Compute $e(shoot, s_1) = \{dead, \neg loaded\}$
2. Eliminate from s_1 those that are false in $e(shoot, s_1)$:
3. Identify maximal sets of atoms X that remain in s_1 and can be joined together with $e(shoot, s_1)$ s.t. $s' = Cn(X \cup e(shoot, s_1))$ is a consistent set of literals and is an interpretation.

States and Transitions: Example

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \end{cases} \quad \begin{cases} \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{cases}$$

$s_1 = \{\neg dead, walking, loaded\}$. Computer $\Phi(shoot, s_1)$

1. Compute $e(shoot, s_1) = \{dead, \neg loaded\}$
2. Eliminate from s_1 those that are false in $e(shoot, s_1)$: remaining from s_1: $\{walking\}$
3. Identify maximal sets of atoms X that remain in s_1 and can be joined together with $e(shoot, s_1)$ s.t. $s' = Cn(X \cup e(shoot, s_1))$ is a consistent set of literals and is an interpretation.

States and Transitions: Example

$$D_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \end{cases} \quad \begin{array}{l} \textit{shoot} \textbf{ executable\_if } \textit{true} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \end{array}$$

$s_1 = \{\neg dead, walking, loaded\}$. Computer $\Phi(shoot, s_1)$

1. Compute $e(shoot, s_1) = \{dead, \neg loaded\}$
2. remaining from s_1: $\{walking\}$
3. There are only two possibilities: $\emptyset$ and $\{walking\}$
   - $X = \{walking\}$ then $Cn(\{walking\} \cup e(shoot, s_1)) = Cn(\{walking, dead, \neg loaded\}) = \{walking, dead, \neg loaded, \neg walking\}$ - this is inconsistent
   - $X = \emptyset$ then $Cn(\emptyset \cup e(shoot, s_1)) = Cn(\{dead, \neg loaded\}) = \{dead, \neg loaded, \neg walking\}$ - this is consistent and complete set of fluent literals in the domain.

Answer: $\Phi(shoot, s_1) = \{\{dead, \neg loaded, \neg walking\}\}$

States and Transitions: Example

$$D'_y = \begin{cases} \textit{shoot} \textbf{ causes } \textit{dead} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ causes } \neg\textit{loaded} \textbf{ if } \textit{loaded} \\ \textit{shoot} \textbf{ executable\_if } \textit{holding\_gun} \\ \textit{alive} \textbf{ if } \neg\textit{dead} \end{cases} \qquad \begin{array}{l} \textit{load} \textbf{ executable\_if } \neg\textit{loaded} \\ \textit{load} \textbf{ causes } \textit{loaded} \\ \neg\textit{walking} \textbf{ if } \textit{dead} \\ \neg\textit{alive} \textbf{ if } \textit{dead} \end{array}$$

$s_1 = \{\textit{holding\_gun}, \neg\textit{dead}, \textit{walking}, \textit{loaded}, \textit{alive}\}$. $\Phi(\textit{shoot}, s_1) = ?$

1. Compute $e(\textit{shoot}, s_1) = \{\textit{dead}, \neg\textit{loaded}\}$
2. remaining from s_1: $\{\textit{alive}, \textit{walking}, \textit{holding\_gun}\}$
3. Identify X remaining in s_1 s.t. $s' = Cn(X \cup e(\textit{shoot}, s_1))$
   If $X = \{\textit{holding\_gun}\}$ then $Cn(\{\textit{holding\_gun}\} \cup e(\textit{shoot}, s_1)) = \{\textit{holding\_gun}, \textit{dead}, \neg\textit{loaded}, \neg\textit{alive}, \neg\textit{walking}\}$ — this is consistent and complete set of fluent literals in the domain. So,

Answer: $\Phi(\textit{shoot}, s_1) = \{\{\textit{dead}, \neg\textit{loaded}, \neg\textit{walking}, \textit{holding\_gun}, \neg\textit{alive}\}\}$

$\mathcal{AL}$: Non-Deterministic Domain

Consider a domain with actions a and b, three fluents f, g, h and the following statements

a, b **executable_if** *true*
a **causes** f
b **causes** $\neg g$ **if** f
$\neg h$ **if** g, f
$\neg g$ **if** h, f

$\Phi(a, \{g, h, \neg f\}) = \{\{f, g, \neg h\}, \{f, h, \neg g\}\}$
This domain is non-deterministic!

$\models$ in $\mathcal{AL}$

How is $\models$ defined in $\mathcal{AL}$?

$\models$ in $\mathcal{AL}$

How is $\models$ defined in $\mathcal{AL}$?

Need to consider $\Phi(a, \omega)$ where ω is a set of states!

- a is executable in ω if a is executable in every $u \in \omega$
  Notation: $\Phi(a, s) = \emptyset$ when a is not executable in s.
- For a set of states ω,

$$\Phi(a, \omega) = \begin{cases} \bigcup_{u \in \omega} \Phi(a, u) \text{ if } \forall u \in \omega.[\Phi(a, u) \neq \emptyset] \\ \\ \emptyset \text{ otherwise} \end{cases}$$

$\models$ in $\mathcal{AL}$

Let $\alpha = [a_1; \ldots; a_n]$ and ω be a set of states.

- $\widehat{\Phi}(\alpha, \emptyset) = \emptyset$
- $\widehat{\Phi}([], \omega) = \omega$
- $\widehat{\Phi}([a, \beta], \omega) = \bigcup_{u \in \Phi(a,\omega)} \widehat{\Phi}(\beta, u)$ if $\widehat{\Phi}(\beta, u) \neq \emptyset$ for every $u \in \widehat{\Phi}(\beta, u)$.

$(D, I) \models I$ **after** α iff $\widehat{\Phi}(\alpha, s_0) \neq \emptyset$ and I is true in every $u \in \widehat{\Phi}(\alpha, s_0)$

Sensing Actions

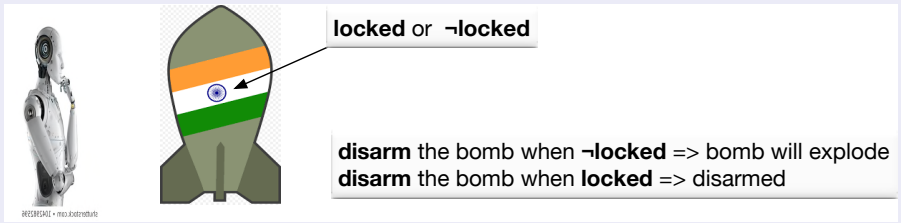
Disarming a Bomb

A robot needs to disarm a bomb. He does not know whether the tip of the bomb was locked or not. Looking at it will help!

Sensing Actions

Disarming a Bomb

A robot needs to disarm a bomb. He does not know whether the tip of the bomb was locked or not. Looking at it will help!



Sensing Actions

1. do not change the state of the world
2. change the beliefs (knowledge) of the reasoner
3. needed for planning and reasoning with incomplete information

Sensing Actions

Disarming a Bomb

A robot needs to disarm a bomb. He does not know whether the tip of the bomb was locked or not. Looking at it will help!

Sensing Actions

1. do not change the state of the world (the robot looking at the tip does not change anything in the physical world!)
2. change the beliefs (knowledge) of the reasoner (the robot looking at the tip changes its knowledge!)
3. needed for planning and reasoning with incomplete information (initially, the robot does not know whether or not the tip is locked).

$\mathcal{AS}$: $\mathcal{AL}$ with Sensing Actions — Syntax

action **determines** *fluent* (5)

Intuition: execution of action allows the reasoner to know the value of fluent.

- *look* **determines** *locked*
- *look_at_departure_screen* **determines** $gate\_of\_flight_X$
  (might use multi-value fluent)

Action Theory

An action theory in $\mathcal{AS}$ is a pair (D, I) where D consists of statements of the form (1)-(2), (4), and (5) and I consists of statements of the form (3).

$\mathcal{AS}$: $\mathcal{AL}$ with Sensing Actions — Semantics

Define the transition function Φ. **What is a state and what is a transition?**

$\mathcal{AS}$: $\mathcal{AL}$ with Sensing Actions — Semantics

Define the transition function Φ. **What is a state and what is a transition?**

Intuition: when an agent has incomplete information

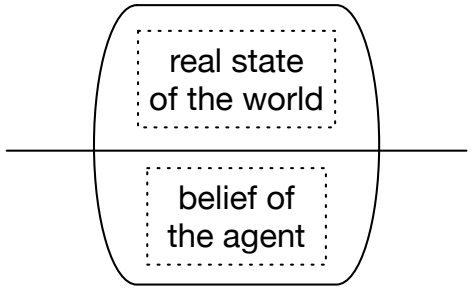
1. its belief consists of a number of possible states that the agent believes it might be in.
2. execution of a sensing action will help the agent to shrink the set of possible states.
3. execution of a non-sensing action will create transition between set of possible states.

$\mathcal{AS}$: $\mathcal{AL}$ with Sensing Actions — Semantics

Define the transition function Φ.

Intuition: when an agent has incomplete information

1. its belief consists of a number of possible states that the agent believes it might be in.
2. execution of a sensing action will help the agent to shrink the set of possible states.
3. execution of a non-sensing action will create transition between set of possible states.

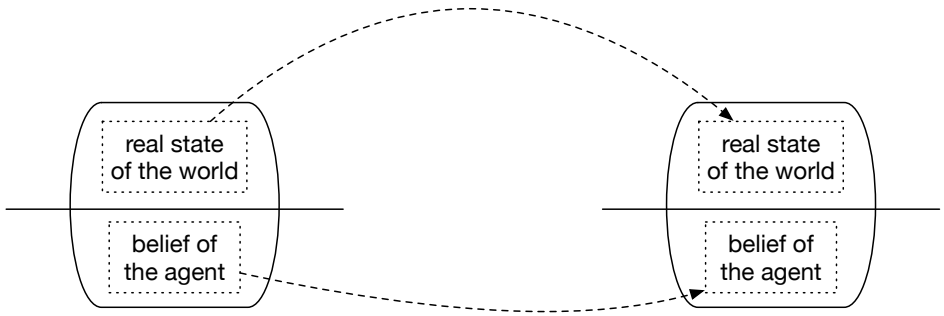


$\mathcal{AS}$: $\mathcal{AL}$ with Sensing Actions — Semantics

Define the transition function Φ.

Intuition: when an agent has incomplete information

1. its belief consists of a number of possible states that the agent believes it might be in.
2. execution of a sensing action will help the agent to shrink the set of possible states.
3. execution of a non-sensing action will create transition between set of possible states.



$\mathcal{AS}$: States and Transitions

For a domain D in $\mathcal{AS}$,

- (*state*): A k-state is a pair $\langle s, \Sigma \rangle$ where s is a state and Σ is a set of states in D.
- $\langle s, \Sigma \rangle$ is consistent if $s \in \Sigma$.
- An action a is executable in $\langle s, \Sigma \rangle$ if it is executable in s.
- φ is known to be true in $\langle s, \Sigma \rangle$ if φ is true in every $u \in \Sigma$.
  - if a is a non-sensing action:
    $\Phi^s(a, \langle s, \Sigma \rangle) = \{\langle s', \Sigma' \rangle \mid s' \in \Phi(a, s), \Sigma' = \bigcup_{u \in \Sigma} \Phi(a, u)\}$ where $\Phi(a, s)$ is defined as in $\mathcal{AL}$ domains
  - if a is a sensing action:
    $\Phi^s(a, \langle s, \Sigma \rangle) = \{\langle s, \Sigma' \rangle \mid \Sigma' = \{u \in \Sigma \mid u \sim_f s\}\}$ where $u \sim_f s$ iff $f \in u \cap s$ or $\neg f \in u \cap s$.

$\mathcal{AS}$: Example

Disarming a Bomb

$$D_b = \begin{cases} \textit{disarm} \textbf{ causes } \textit{exploded} \textbf{ if } \neg \textit{locked} \\ \textit{disarm} \textbf{ causes } \textit{disarmed} \textbf{ if } \textit{locked} \\ \textit{disarm} \textbf{ causes } \neg \textit{exploded} \textbf{ if } \textit{locked} \\ \textit{turn} \textbf{ causes } \neg \textit{locked} \textbf{ if } \textit{locked} \\ \textit{turn} \textbf{ causes } \textit{locked} \textbf{ if } \neg \textit{locked} \\ \textit{disarmed} \textbf{ if } \textit{exploded} \\ \neg \textit{locked} \textbf{ if } \textit{exploded} \\ \textit{look} \textbf{ determines } \textit{locked} \end{cases}$$

$$I_b = \begin{cases} \textbf{initially } \neg \textit{disarmed} \\ \textbf{initially } \neg \textit{exploded} \\ \\ \\ \textit{locked} \text{ unknown} \end{cases}$$

Goal: $\textit{disarmed}, \neg \textit{exploded}$

$\models$ in $\mathcal{AS}$

Define $\widehat{\Phi}$ in the same way as in $\mathcal{AL}$ but need branches

Conditional Plan

1. An empty sequence of action, denoted by [ ], is a conditional plan.
2. If a is an action then a is a conditional plan.
3. If $\alpha_1, \ldots, \alpha_n$ are conditional plans and φ_j's are mutual exclusive conjunctions of fluent literals then the following is a conditional plan.

   Case

   $\varphi_1 \rightarrow \alpha_1$

   $\ldots$

   $\varphi_n \rightarrow \alpha_n$

   Endcase
4. If α_1 and α_2 are conditional plans then $\alpha_1; \alpha_2$ is a conditional plan.
5. Nothing else is a conditional plan.

$\models$ in $\mathcal{AS}$

For a set of states ω, and a case plan α:

$\alpha =$ **Case**

$\varphi_1 \rightarrow \alpha_1$

$\ldots$

$\varphi_n \rightarrow \alpha_n$

Endcase

$$\hat{\Phi}(\alpha, \sigma) = \begin{cases} \hat{\Phi}(\alpha_i, \sigma) \text{ if } \varphi_i \text{ is known to be true in } \sigma \\ \\ \emptyset \text{ if there exists no } i \text{ s.t. } \varphi_i \text{ is known to be true in } \sigma \end{cases}$$

$\mathcal{AS}$: Example

The set of states of D_b:

$s_1 = \{\neg disarmed, \neg locked, \neg exploded\}$ $\quad$ $s_3 = \{disarmed, \neg locked, \neg exploded\}$
$s_2 = \{\neg disarmed, locked, \neg exploded\}$ $\quad$ $s_4 = \{disarmed, exploded, \neg locked\}$
$s_5 = \{disarmed, locked, \neg exploded\}$ $\quad$ $s_6 = \{disarmed, locked, exploded\}$
$s_7 = \{exploded, \neg disarmed, \neg locked\}$ $\quad$ $s_8 = \{locked, exploded, \neg disarmed\}$

$\Sigma_0 = \{s_1, s_2\}$ — two initial k-states representing the beliefs of the robot:

$$\langle s_1, \Sigma_0 \rangle \quad \text{and} \quad \langle s_2, \Sigma_0 \rangle$$

execution of look in the initial k-state results in

$$\langle s_1, \{s_1\} \rangle \quad \text{and} \quad \langle s_2, \{s_2\} \rangle$$

which means

- if the real state of the world is s_1, then execution *look* will help the robot knows that the tip is $\neg locked$.
- if the real state of the world is s_2, then execution *look* will help the robot knows that the tip is *locked*.

Non-Deterministic Actions

Example

Flipping a coin results in head or tail ($\neg head$).

$$action \textbf{ maychange } literal_1 \mid \cdots \mid literal_n \tag{6}$$

Intuition: execution of action results in one of the possibilities $literal_1, \ldots, literal_n$ with the assumption that $literal_1, \ldots, literal_n$ are mutual exclusive.

- $flip$ **maychange** $head \mid tail$
- $shoot$ **maychange** $dead \mid alive$

$\mathcal{AS}^n$: $\mathcal{AS}$ with Non-Deterministic Actions

What is a state?

What is a transition?

$\mathcal{AS}^n$: $\mathcal{AS}$ with Non-Deterministic Actions

What is a state? Use k-state as in $\mathcal{AS}$

What is a transition?

$\mathcal{AS}^n$: $\mathcal{AS}$ with Non-Deterministic Actions

What is a state? Use k-state as in $\mathcal{AS}$

What is a transition? Define $\Phi^s(a, s)$.

- Need only to specify transitions for non-deterministic actions!
- Below: theories without static causal laws.
- For theories with static causal laws, similar adaptation is needed.

$\mathcal{AS}^n$: $\mathcal{AS}$ with Non-Deterministic Actions

What is a state? Use k-state as in $\mathcal{AS}$

What is a transition? Define $\Phi^s(a, s)$.

- Need only to specify transitions for non-deterministic actions!
- Below: theories without static causal laws.
- For theories with static causal laws, similar adaptation is needed.

Assume that

$$a \textbf{ maychange } l_1 \mid \cdots \mid l_n$$

Define $e_i = l_i \cup \{\overline{l_j} \mid j \neq i, 1 \leq j \leq n\}$ ($\bar{l}$ is the negation of l).
For each state s, let $\Omega(s) = \{s \setminus \overline{e_i} \cup e_i \mid i = 1, \ldots, n\}$.

- if a is executable in $\langle s, \Sigma \rangle$ then

$$\Phi^s(a, \langle s, \Sigma \rangle) = \{\langle s \setminus \overline{e_i} \cup e_i, \bigcup_{u \in \Sigma} \Omega(u) \rangle \mid i = 1, \ldots, n\}$$

  A k-state in $\Phi^s(a, \langle s, \Sigma \rangle)$: represents a possibility.
- otherwise, $\Phi^s(a, \langle s, \Sigma \rangle) = \emptyset$.

$\mathcal{AS}^n$: Example

Consider the action

$$\textit{flip} \textbf{ maychange } \textit{head} \mid \neg \textit{head}$$

and the current state of the world $s = \{head\}$.
Assume that the agent has complete information about the world.
The k-state representing this situation is $\langle \{head\}, \{\{head\}\} \rangle$.
Execution of *flip* in this k-state results to

$$\{\langle \{head\}, \{\{head\}, \{\neg head\}\} \rangle, \langle \{\neg head\}, \{\{head\}, \{\neg head\}\} \rangle\}$$

$\models$ in $\mathcal{AS}^n$

Defined as in $\mathcal{AS}$.

GOLOG

GOLOG: logic programming language for dynamic domains [Levesque et al. (1997)].

Motivations:

- Ease the development of control programs of dynamic domains (high level controllers for robots, intelligent software agents, etc.)
- Provide a means for reasoning about complex actions.

Note: The language has been extended to ConGolog [De Giacomo et al. (2000)] to allow concurrency, interrupts, concurrency with priorities, and concurrent iteration.

GOLOG: Syntax

Action theories are written in a sorted first-order language with finite domains, e.g., in the **block domain**, *block* is a sort with domain, say, $\{a, b, c\}$; *on* is a binary fluent of the sort $block \times block$; and *pickup* is an unary action of the sort *block*, etc.

Program: a is an action, ϕ is a formula p and q are programs

- Primitive: a
- Test: $\phi?$
- Sequence: $p; q$
- Non-deterministic choice of actions: $p \mid q$
- Conditional: **if** ϕ **then** p **else** q **endif**
- While-loop: **while** ϕ **then** p **endWhile**
- Procedure: **proc** p **endProc**
- Non-deterministic choice of arguments: $\pi(X, p)$

are programs.

GOLOG: Example

Consider the block world domain with

- constants (of the sort block): a, b, c, d, e
- actions such as $putdown(x)$, $pickup(x)$, $stack(x, y)$, $unstack(x, y)$, etc.
- fluents such as $onTable(x)$, $on(x, y)$, $clear(x)$, etc.

Some GOLOG programs

$make\_on(x, y)$: $makeClear(x)$;
if $onTable(x) \wedge clear(y)$
then $[pickup(x); stack(x, y)]$
else $[makeClear(y); get(x);$
$stack(x, y)]$
endif

$makeClear(x)$: $clear(x)? \mid \neg clear(x)?$
$cleanHand$;
while $\neg clear(x)$ **then**
$\pi(y, z, [(clear(y) \wedge$
$above(z, x) \wedge on(y, z))?;$
$unstack(y, z); putdown(y)])$
endWhile

- $cleanHand$: **if** $\neg handEmpty$ **then** $\pi(x, [holding(x)?; putdown(x)])$ **endif**
- $get(x)$: **if** $onTable(x)$ **then** $pickup(x)$
else $\pi(y, [on(x, y)?, unstack(x, y)])$ **endif**

GOLOG: Another Example

$D_{elevator}$ consists of

$up(N)$ **causes** $cFloor(N)$
$down(N)$ **causes** $cFloor(N)$
$turnoff(N)$ **causes** $\neg on(N)$
$open$ **causes** $opened$
$close$ **causes** $\neg opened$
$cFloor(M)$ **if** $\neg cFloor(N)$

$up(N)$ **executable_if** $cFloor(M), \neg opened\ (M < N)$
$down(N)$ **executable_if** $cFloor(M), \neg opened\ (M > N)$
$turnoff(N)$ **executable_if** $cFloor(N)$

$$S = \left\{ \begin{array}{rcl} (go\_floor(N) & : & cFloor(N) | up(N) | down(N)) \\ (serve(N) & : & go\_floor(N); turnoff(N); open; close) \\ (serve\_a\_floor & : & \pi(N, (on(N)?; serve(N))) \\ (park & : & \textbf{if}\ \ cFloor(0)\ \textbf{then}\ \ open\ \textbf{else}\ \ [down(0); open]) \\ (control & : & [\textbf{while}\ \exists N.\{0, \ldots, k\}\ [on(N)]\ \textbf{do}\ \ serve\_a\_floor]; park) \end{array} \right.$$

GOLOG: Semantics

GOLOG programs are interpreted with respect to traces of the form $s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n$ where $s_{i+1} \in \Phi(a, s_i)$ for every $i = 0, \ldots, n-1$.

$s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n$ is a trace for a GOLOG program δ if

- $\delta = a$ and a is an action, $n = 1$ and $a_0 = a$;
- $\delta = \phi?$, $n = 0$ and ϕ holds in s_0;
- $\delta = \delta_1; \delta_2$, and there exists an i such that $s_0 a_0 \ldots s_i$ is a trace of δ_1 and $s_i a_i \ldots s_n$ is a trace of δ_2;
- $\delta = \delta_1 \mid \delta_2$, and $s_0 a_0 \ldots a_{n-1} s_n$ is a trace of δ_1 or δ_2;
- $\delta =$ **if** ϕ **then** δ_1 **else** δ_2 **endif**, and $s_0 a_0 \ldots a_{n-1} s_n$ is a trace of δ_1 if ϕ holds in s_0 or $s_0 a_0 \ldots a_{n-1} s_n$ is a trace of δ_2 if $\neg\phi$ holds in s_0;
- ...

Action Languages and Related Approaches

Other approaches to reasoning about actions and changes:

- situation calculus [McCarthy and Hayes (1969)]
- event calculus [Kowalski and Sergot (1986)]
- fluent calculus [Thielscher (2000)]
- STRIPS [Fikes and Nilson (1971)]
- PDDL [Ghallab et al. (1998)]: de facto language for planning systems

$\mathcal{AL}$ vs. PDDL (mostly a 1-1 correspondence, difference in static causal laws)

Driving to the airport domain in PDDL representation

```
(define (domain airport)
    (:predicates (at ?x ?y)
       (location ?x) (person ?p) (car ?c))
    (:action drive :parameters (?x ?y)
       :precondition (and (location ?x) (location ?y)
          (person ?p) (at ?p ?x) (car ?c) (at ?c ?x))
       :effect (and (at ?c ?y) (at ?p ?y)
          (not (at ?c ?x)) (not (at ?p ?x)))))
```

Problem: δ_a and Goal in PDDL representation

```
(define (problem airport-1-1) (:domain airport)
(:objects john car home airport)
(:init person(john) car(car) location(home) location(airport)
       at(john,home) at (car,home))
(:goal at(john,airport)))
```

$\mathcal{AL}$ vs. PDDL (mostly a 1-1 correspondence, difference in static causal laws)

Driving to the airport domain in PDDL representation

```
(define (domain airport)
    (:predicates (at ?x ?y)
       (location ?x) (person ?p) (car ?c))
    (:action drive :parameters (?x ?y)
       :precondition (and (location ?x) (location ?y)
          (person ?p) (at ?p ?x) (car ?c) (at ?c ?x))
       :effect (and (at ?c ?y) (at ?p ?y)
          (not (at ?c ?x)) (not (at ?p ?x)))))
```

Proble

```
(defin
(:obje
(:init
(:goal
```

in $\mathcal{AL}$

$drive(home, airport)$ **executable_if** $at(john, home), at(car, john)$
$drive(home, airport)$ **causes** $at(john, airport), at(car, airport)$
$drive(airport, home)$ **executable_if** $at(john, airport), at(car, airport)$
$drive(airport, home)$ **causes** $at(john, home), at(car, home)$
$\neg at(john, airport)$ **if** $at(john, home)$
$\neg at(car, airport)$ **if** $at(car, home)$
$\neg at(john, home)$ **if** $at(john, airport)$
$\neg at(car, home)$ **if** $at(car, airport)$

$\mathcal{AL}$ vs. PDDL (mostly a 1-1 correspondence, difference in static causal laws)

Driving to the airport domain in PDDL representation

```
(define (domain airport)
    (:predicates (at ?x ?y)
       (location ?x) (person ?p) (car ?c))
    (:action drive :parameters (?x ?y)
```

in $\mathcal{AL}$

initially $at(john, home)$
initially $at(car, home)$
...

Problem: δ_a and Goal in PDDL representation

```
(define (problem airport-1-1) (:domain airport)
(:objects john car home airport)
(:init person(john) car(car) location(home) location(airport)
       at(john,home) at (car,home))
(:goal at(john,airport)))
```

$\mathcal{AL}$ vs. PDDL

$\mathcal{AL}$	PDDL
Action	$\surd$
Fluent	Predicate
Conditional Effect	$\surd$
Executability condition	Precondition
Static causal law (allow cyclic)	Defined fluent or axiom (no cyclic)
Ground Instantiations (Variables: shorthand)	Typed Variables

$\mathcal{AL}$ vs. PDDL

$\mathcal{AL}$	PDDL
Action	$\surd$
Fluent	Predicate
Conditional Effect	$\surd$
Executability condition	Precondition
Static causal law (allow cyclic)	Defined fluent or axiom (no cyclic)
Ground Instantiations (Variables: shorthand)	Typed Variables

Notes

1. Dealing directly with static causal laws is advantageous [Thiebaux et al. (2003)].
2. Not many planners deal with static causal laws directly.

$\mathcal{AL}$ vs. PDDL

Example of cyclic static causal laws in $\mathcal{AL}$:

- A door is either closed or opened:

$$\textit{door\_opened} \textbf{ if } \neg \textit{door\_closed}$$
$$\textit{door\_closed} \textbf{ if } \neg \textit{door\_opened}$$

- John is either at home or his office:

$$\textit{at\_home} \textbf{ if } \neg \textit{at\_office}$$
$$\textit{at\_office} \textbf{ if } \neg \textit{at\_home}$$

1. Defined fluents are often not allowed to occur in effects of actions in some PDDL specifications.
2. PDDL has been extended with other features (e.g., sensing actions, actions with durations)

1. Overview: a brief history of action languages (20 mnt - Chitta)
2. Action languages in single agent environments (70 mnt - Son)
   - The action language $\mathcal{A}$, state, and transition function
   - $\mathcal{AL}$: $\mathcal{A}+$ static causal laws, non-deterministic and sensing actions
   - GOLOG
   - Action languages: related approaches and planning
3. Action languages and causality (30 mnt - Chitta)
   - Pearl's do-calculus
4. Action languages in multi-agent environments (60 mnt – Son)
   - mA*, Kripke structure, update models, and transition function
   - mA* in epistemic planning
5. Action languages in commonsense reasoning (18 mnt - Chitta)
   - Commonsense reasoning and actions
   - Acquiring knowledge about actions
6. Open challenges, conclusion, and discussion (12 mnt - Chitta)

Excerpts from Book of Why?

Judea Pearl, Dana Mackenzie - The book of why_ the new science of cause and effect-Basic Books (2018).pdf (page 15 of 402)

Side by side with this diagrammatic "language of knowledge," we also have a symbolic "language of queries" to express the questions we want answers to. For example, if we are interested in the effect of a drug (D) on lifespan (L), then our query might be written symbolically as: $P(L \mid do(D))$. In other words, what is the probability (P) that a typical patient would survive L years if made to take the drug? This question describes what epidemiologists would call an *intervention* or a *treatment* and corresponds to what we measure in a clinical trial. In many cases we may also wish to compare $P(L \mid do(D))$ with $P(L \mid do(\text{not-}D))$; the latter describes patients denied treatment, also called the "control" patients. The *do*-operator signifies that we are dealing with an intervention rather than a passive observation; classical statistics has nothing remotely similar to this operator.

Excerpts from Book of Why?

Mathematically, we write the observed frequency of Lifespan L among patients who voluntarily take the drug as $P(L \mid D)$, which is the standard conditional probability used in statistical textbooks. This expression stands for the probability (P) of Lifespan L conditional on seeing the patient take Drug D. Note that $P(L \mid D)$ may be totally different from $P(L \mid do(D))$. This difference between seeing and doing is fundamental and explains why we do not regard the falling barometer to be a cause of the coming storm. Seeing the barometer fall increases the probability of the storm, while forcing it to fall does not affect this probability.

This confusion between seeing and doing has resulted in a fountain of paradoxes, some of which we will entertain in this book. A world devoid of $P(L \mid do(D))$ and governed solely by $P(L \mid D)$ would be a strange one indeed. For example, patients would avoid going to the doctor to reduce the probability of being seriously ill; cities would dismiss their firefighters to reduce the incidence of fires; doctors would recommend a drug to male and female patients but not to patients with undisclosed gender; and so on. It is hard to believe that less than three decades ago science did operate in such a world: the *do*-operator did not exist.

Simpson's Paradox

The following data is given

Gender	Action	Recovered?	Did not recover	Probability of recovery
Male	Took Drug	18	12	0.6
Male	Did not take drug	7	3	0.7
Female	Took Drug	2	8	0.2
Female	Did not take drug	9	21	0.3

Based on the probability calculated you can say that both male and female, **the probability of recovery is high if they did not take the drug**.

Simpson's Paradox

Based on the probability calculated you can say that both male and female, **the probability of recovery is high if they did not take the drug**.

Now if we did not have the gender data, we would have to reconstruct the table.

Action	Recovered?	Did not recover	Probability of recovery
Took Drug	20	20	0.5
Did not take drug	16	24	0.4

Based on the new probability calculated you can say that both male and female, **the probability of recovery is high if they took the drug**.
So the outcome seems to change based on us knowing the gender of the collected statistics.

Shortcoming of traditional conditional probability

X (Treatment)	Y (Recovered)	Probability
T	T	0.25
T	F	0.25
F	T	0.25
F	F	0.25

Consider the above joint probability distribution.

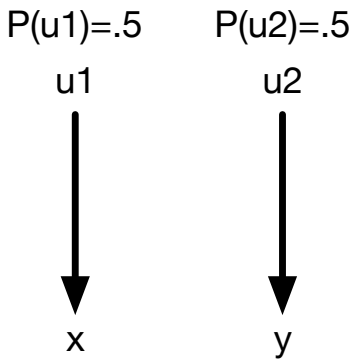
- We can calculate $P(Recovery \mid Treatment)$.
- But is that the right way to decided whether treatment should be given or not.
- Pearl proposes to use $P(recover \mid \mathbf{do}(given\ treatment))$.

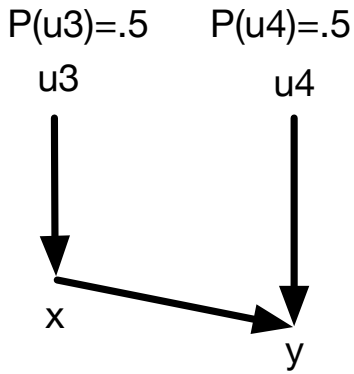
We will show how that value may be different depending on what kind of causal model leads to the above probability distribution.

Probabilistic Causal Model

- There are two kinds of variables in this model:
  - Exogenous ($Ex : u1, u2$): external to the system
  - Endogenous ($Ex : x, y$): internal to the system
- Probabilistic Causal Model is a directed acyclic graph. The value of a node in this graph is defined by a function whose input are its parents. Exogenous variables have no parents and have a probability associated with each of them. All exogenous variables are considered independent of each other.

Two Causal Models: X (given treatment, Y (recovered)





u1 and u2 are variables that determine x and y

x = u1; y = u2

u1	u2	x	y	prob
T	T	T	T	0.25
T	F	T	F	0.25
F	T	F	T	0.25
F	F	F	F	0.25

x = u3

y = (x & u4) or (not x & not u4)

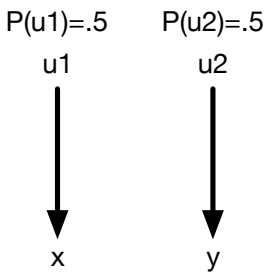
Note: u4 may be some generic condition

u3	u4	x	y	prob
T	T	T	T	0.25
T	F	T	F	0.25
F	T	F	T	0.25
F	F	F	F	0.25

Joe took the treatment and did not survive

"Did the treatment cause this"

(Had he not taken the treatment would he have survived)



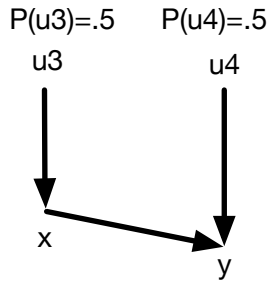
u1	u2	x	y	prob
T	T	T	T	0.25
T	F	T	F	0.25
F	T	F	T	0.25
F	F	F	F	0.25

wrt. new model

T	F	X=F	F	?

P(Y=T | **do**(X=F))=0

Treatment does not cause death.



u3	u4	x	y	prob
T	T	T	T	0.25
T	F	T	F	0.25
F	T	F	T	0.25
F	F	F	F	0.25

new model

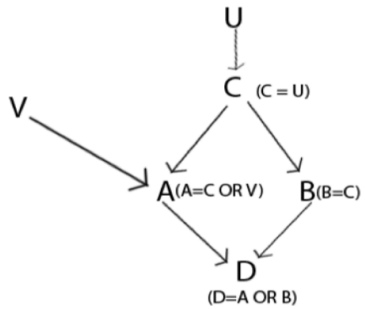
T	F	X=F	T	?

P(Y=T | **do**(X=F))=1

Treatment caused death.

Rifleman Example: actions and counterfactuals

We observe that the prisoner died. What is the probability that the prisoner would be alive if A (on his own) did not shoot.



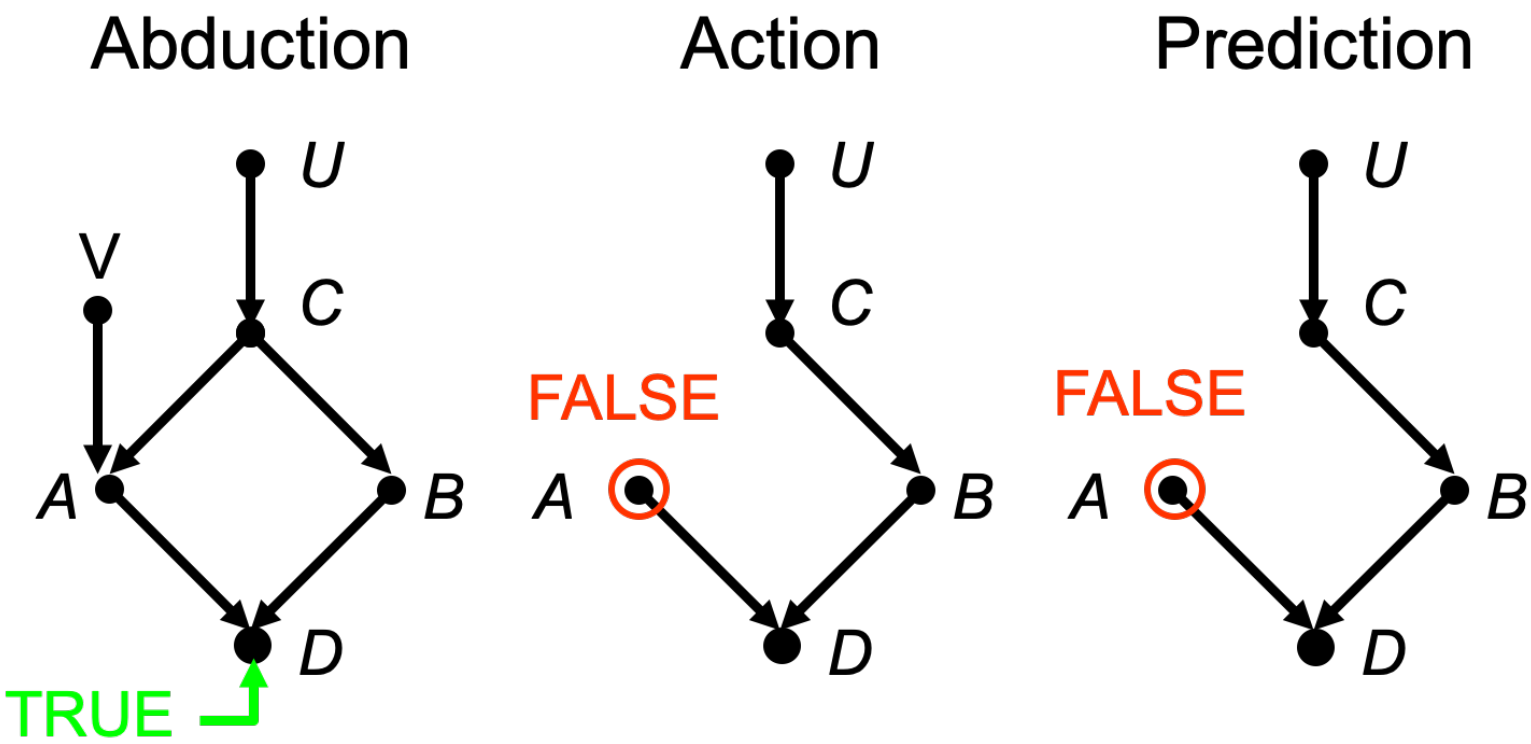
U	V	C	A	B	D	prob
T	T	T	T	T	T	p*q
T	F	T	T	T	T	p*(1-q)
F	T	F	T	F	T	(1-p)*q
F	F	F	F	F	F	(1-p)*(1-q)

Based on the observation we need to remove the last Row and readjust the probabilities: Let S = 1-(1-p)*(1-q)

U	V	C	A = F	B	D	prob
T	T	T	F	T	T	p*q/S
T	F	T	F	T	T	p*(1-q)/S
F	T	F	F	F	F	(1-p)*q/S

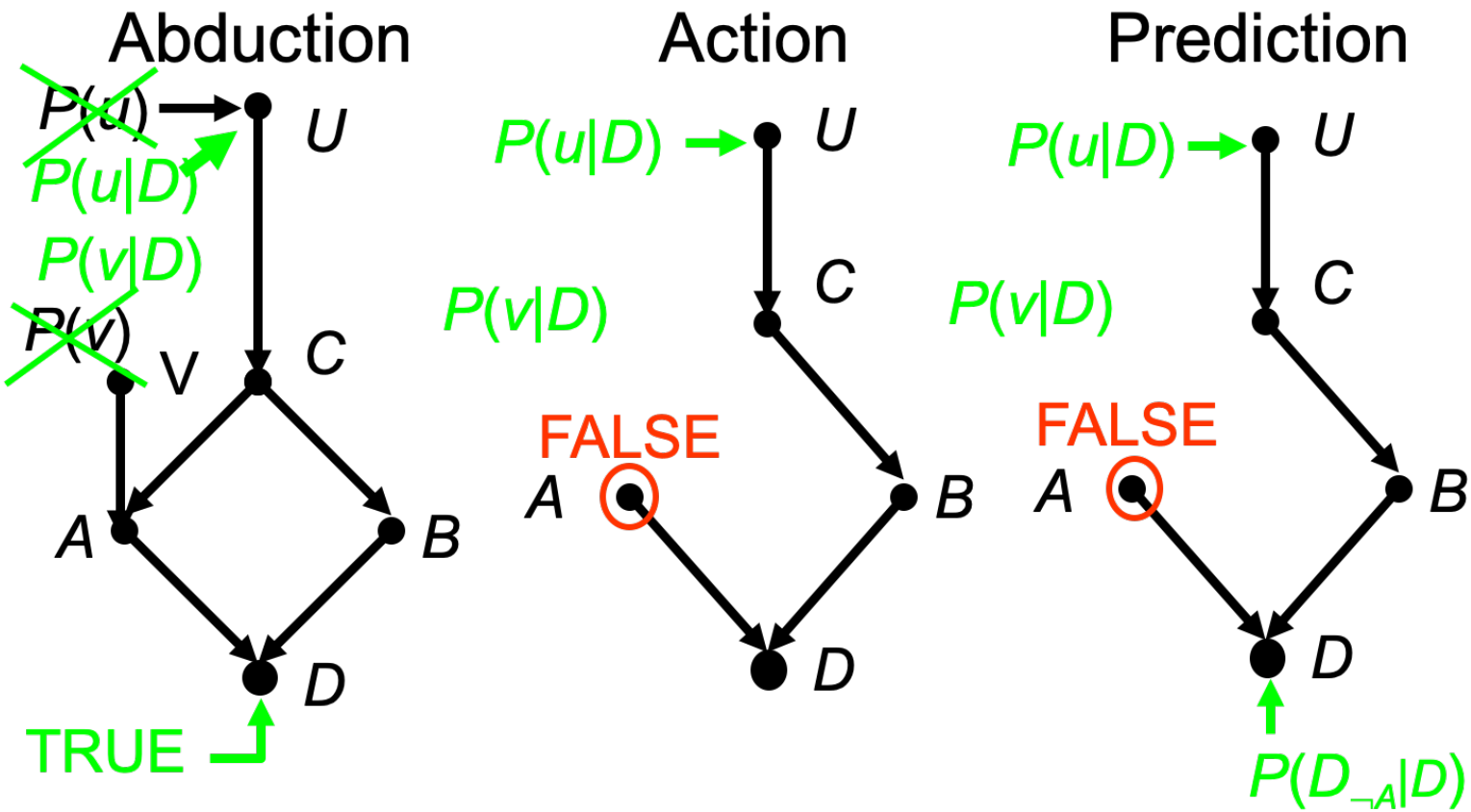
- U, V are exogenous: p(U) = P; p (V) = q
- A, B, C, D are endogenous
- U: Court orders execution
- V: Rifleman A is nervous
- C: Captain give order
- A: Rifleman A shoots
- B: Rifleman B shoots
- D: Prisoner dies

3 Steps to Computing Counterfactuals



Computing Probabilities of Counterfactuals

The prisoner is dead. How likely is it that he would be dead if A had not shot. $P(D_{-A} \mid D) = ?$



Causal model (Formal): From Pearl's Slides

Causal model

$M = \langle U, V, F \rangle$ or $\langle U, V, F, P(u) \rangle$

- U - background variables
- V - endogenous variables
- F - set of functions $\{U \times V \setminus V_i \rightarrow V_i\}$

$$v_i = f_i(pa_i, u_i)$$

Submodel

$M_x = \langle U, V, F_x \rangle$ representing do(x)

F_x = replaces equation for X with X=x

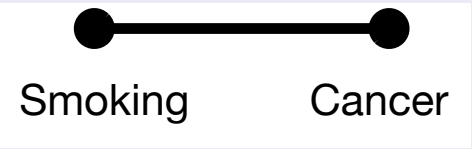
Actions and Counterfactuals

$Y_x(u)$ = solution of Y in M_x

$P(y \mid do(x))$ $\quad$ $P(Y_x = y)$

Predicting the Effects of Policies

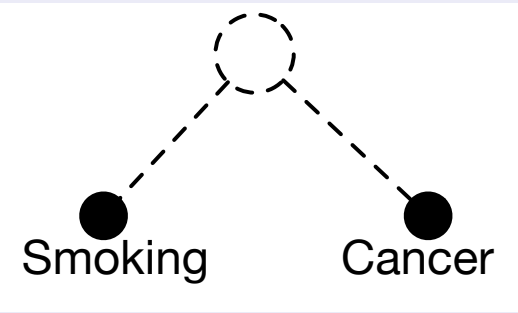
Surgeon General (1964):



$$P(c \mid do(s)) \sim P(c \mid s)$$

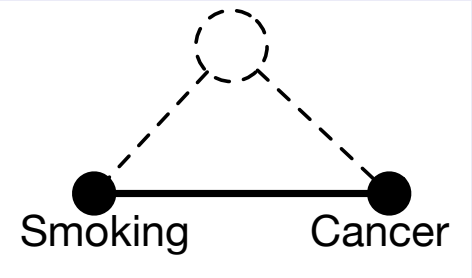
Tobacco Industry:

Genotype (unobserved)



$$P(c \mid do(s)) = P(c)$$

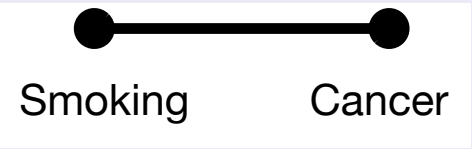
Combined



$$P(c \mid do(s)) = \text{noncomputable}$$

Predicting the Effects of Policies

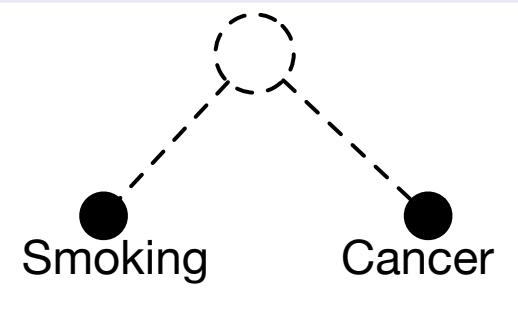
Surgeon General (1964):



$$P(c \mid do(s)) \sim P(c \mid s)$$

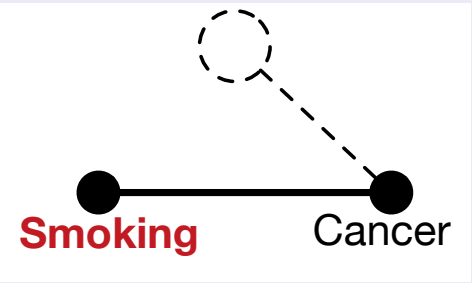
Tobacco Industry:

Genotype (unobserved)



$$P(c \mid do(s)) = P(c)$$

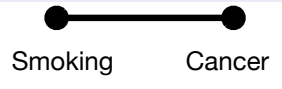
Combined



$$P(c \mid do(s)) = \text{noncomputable}$$

Predicting the Effects of Policies

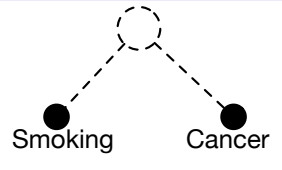
Surgeon General (1964):



$$P(c \mid do(s)) \sim P(c \mid s)$$

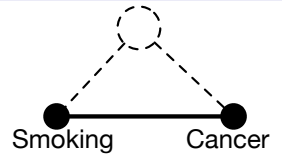
Tobacco Industry:

Genotype (unobserved)



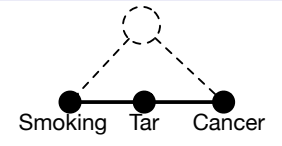
$$P(c \mid do(s)) = P(c)$$

Combined



$$P(c \mid do(s)) = \text{noncomputable}$$

Combined and refined



$$P(c \mid do(s)) = \text{computable}$$

Needed: Algebra of Doing

Available: algebra of seeing

e.g., what is the chance it rained if we see the grass wet?

$P(rain \mid wet) =? \{P(wet \mid rain)\frac{P(rain)}{P(wet)}\}$

Needed: algebra of doing

e.g., what is he chance it rained if we make the grass wet?

$P(rain \mid do(wet)) =? \{P(rain)\}$

Rules of Causal Calculus

- Rule 1: Ignoring observations (under conditions)

  $P(y \mid do\{x\}, z, w) = P(y \mid do\{x\}, w)$

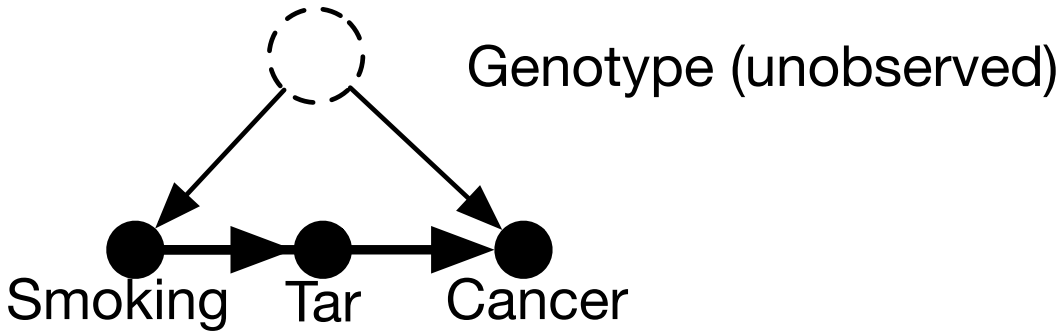
- Rule 2: Action/observation exchange (under conditions)

  $P(y \mid do\{x\}, do\{z\}, w) = P(y \mid do\{x\}, z, w)$

- Rule 3: Ignoring actions (under conditions)

  $P(y \mid do\{x\}, do\{z\}, w) = P(y \mid do\{x\}, w)$

Derivation in Causal Calculus



$$P(c \mid do\{s\}) = \Sigma_t P(c \mid do\{s\}, t) P(t \mid do\{s\}) \quad \text{Probability Axioms}$$

$$= \Sigma_t P(c \mid do\{s\}, do\{t\}) P(t \mid do\{s\}) \quad \text{Rule 2}$$

$$= \Sigma_t P(c \mid do\{s\}, do\{t\}) P(t \mid s) \quad \text{Rule 2}$$

$$= \Sigma_t P(c \mid do\{t\}) P(t \mid s) \quad \text{Rule 3}$$

$$= \Sigma_{s'} \Sigma_t P(c \mid do\{t\}, s') P(s' \mid do\{t\}) P(t \mid s) \quad \text{Probability Axioms}$$

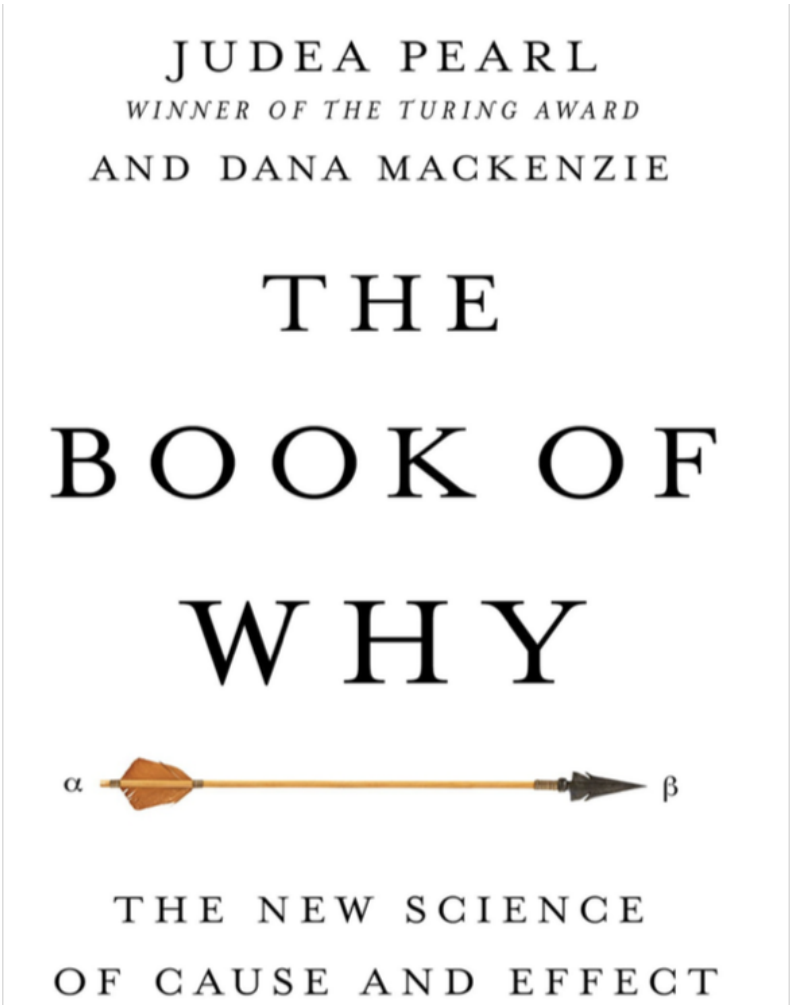
$$= \Sigma_{s'} \Sigma_t P(c \mid t, s') P(s' \mid do\{t\}) P(t \mid s) \quad \text{Rule 2}$$

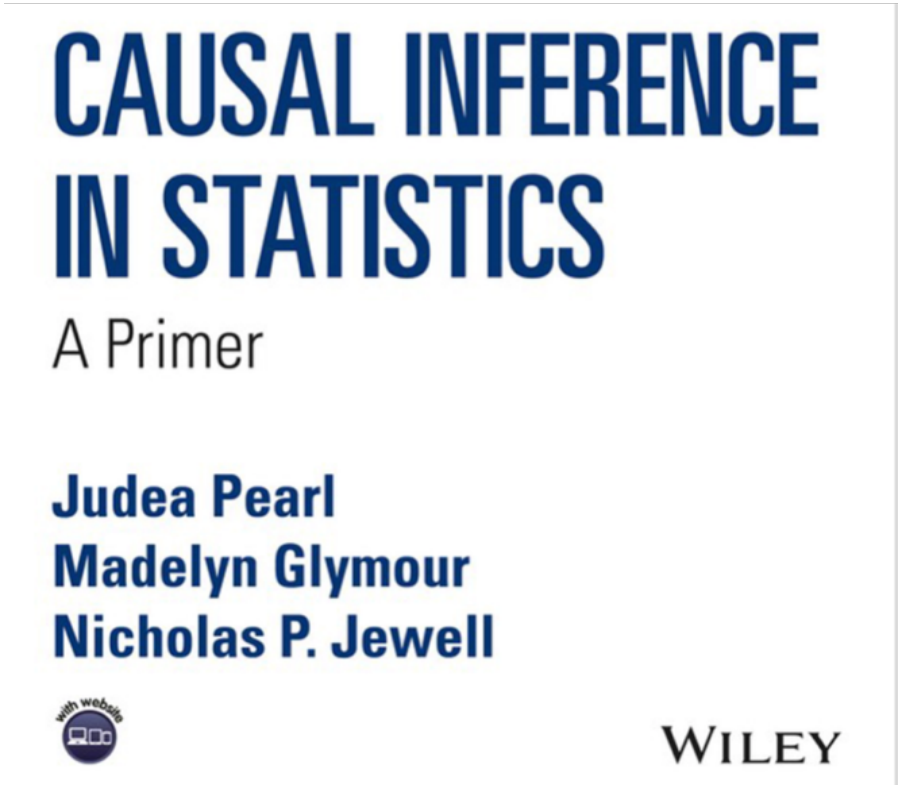
$$= \Sigma_{s'} \Sigma_t P(c \mid t, s') P(s') P(t \mid s) \quad \text{Rule 3}$$

Two directions of research

- Actual cause—Halpern and Pearl; and then many others
- Reasoning about cause and effect from Statistical data—Pearl's recent BIG focus

To recent books by Pearl





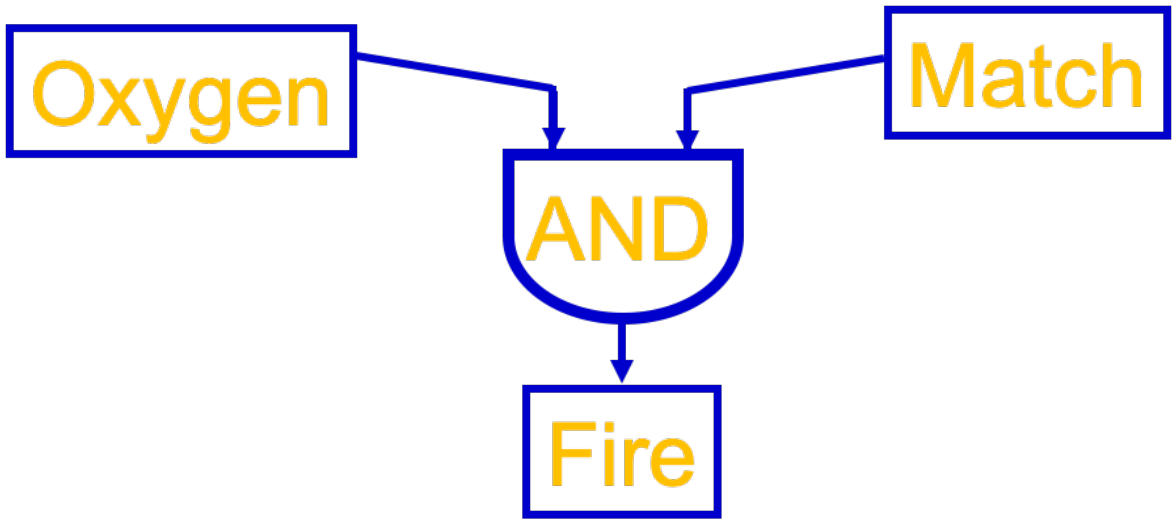
Actual Causes

- **Simple definition:** Suppose f is observed, we can infer that a is an actual cause of f if f would not be true if a had not been true.
- **Motivation:**
  - Hume, Enquiry, 1748: "We may define a cause to be an object followed by another, . . . , where, if the first object had not been, the second never had existed."
  - Lewis (1973): "x CAUSED y" if x and y are true, and y is false in the closest non-x-world.
- Problem with the simple definition:
  - NECESSITY
    - ★ Ignores aspects of sufficiency (Production)
    - ★ Fails in presence of other causes (Over determination)

    COARSENESS
    - ★ Ignores structure of intervening mechanisms.
    - ★ Fails when other causes are preempted (Preemption)

Pearl: Match is the cause here

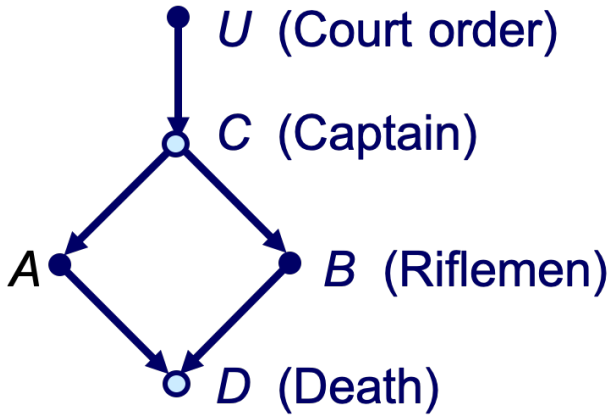
Sufficiency (Production)



Observation: Fire broke out.
Question: Why is oxygen an awkward explanation?
Answer: Because Oxygen is (usually) not sufficient
P(Oxygen is sufficient) = P(Match is lighted) = low
P(Match is sufficient) = P(Oxygen present) = high

Pearl: overdetermination

How the counterfactual test fails?

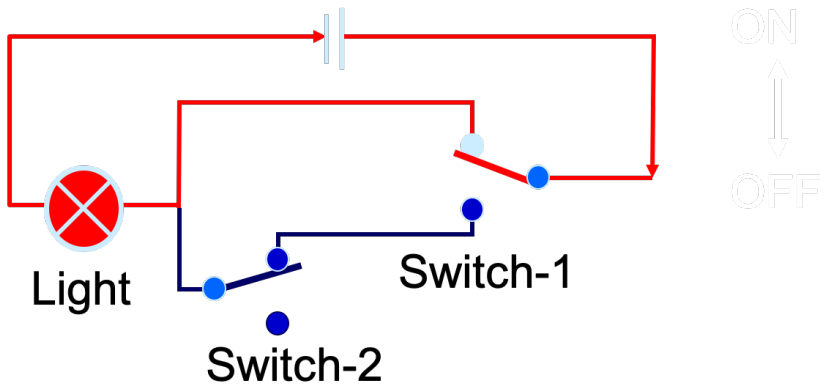


Observation: Dead prisoner with two bullets.
Query: Was A a cause of death?
Answer: Yes, A sustains D against B.

Pearl: preemption

How the counterfactual test fails?

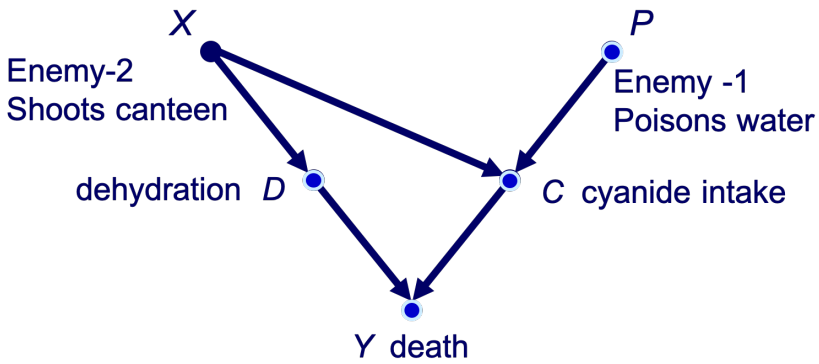
Which switch is the actual cause of light? S1!



Deceiving symmetry: Light = S1 $\vee$ S2

The desert traveler: Enemy2 is the cause

(Pearl; Pat Suppes)



Challenges to Halpern & Pearl (1)

- Batusov & Soutchanski (2018): "The ontological commitments of structural causal models resemble propositional logic, they have no objects, no relationships, no time, no support for quantified causal queries. Thus, causal models are too coarse to distinguish between enduring conditions and transitional events, providing only atomic propositions to model both.
- Moreover, causal models represent presence and absence of an event identically—by assigning a value to a propositional variable. Both of these deficiencies stem from the lack of a mechanism for modeling change over time."
- "In contrast to HP whose analysis is based on observing the end results of interventions, we do so by analyzing the dynamics which lead to the end results."

Challenges to Halpern & Pearl (2)

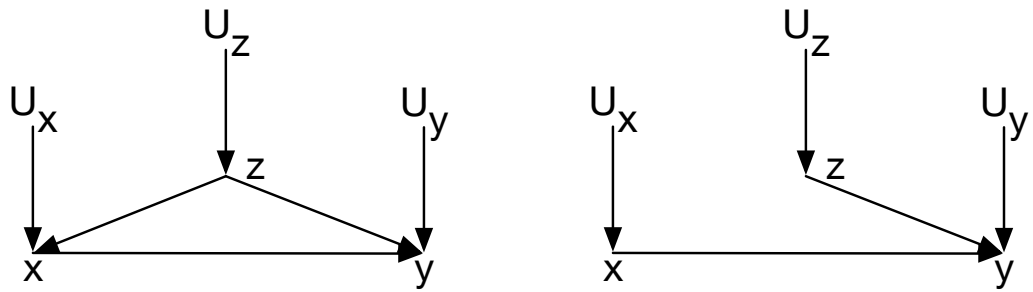
(Beckers and Vennekens 2012)

- Assassin poisons Victims coffee, Victim drinks it and dies. If Assassin hadn't poisoned the coffee, Backup would have, and Victim would have died anyway. Victim would not have died if there had been no poison in the coffee.
  - HP designates Assassin as the actual cause of Dies.
- An engineer is standing by a switch in the railroad track. A train approaches in the distance. She flips the switch, so that the train travels down the left-hand track, instead of the right. Since the tracks reconverge up ahead, the train arrives at its destination all the same.
  - HP designates the flipping of the switch as a cause of the train arriving at its destination.
  - B & V feel that directing the train from one track to another that serves exactly the same purpose is not a cause of its arrival.

Reasoning about cause and effect from Statistical data

- **Basic idea:** How to reason about $P(X \mid do(Y))$ when we have only statistical data
- Pearl proposes ways to do it if causal relations between X, Y and other related variables follow certain patterns.
  - Note: The exact functions connecting the variables need not be known. Only the arrows between the variables depicting the causal connections is enough.
- An example in the next few slides.

Example



Graphical model on effects of new drug: Z-gender, X-drug usage, Y-recovery (Right: sets the drug usage in the population, results in the manipulated probability P_m)

$$P(Y = y \mid do(X = x)) = \Sigma_z P(Y = y \mid X = x, Z = z)P(Z = z)$$

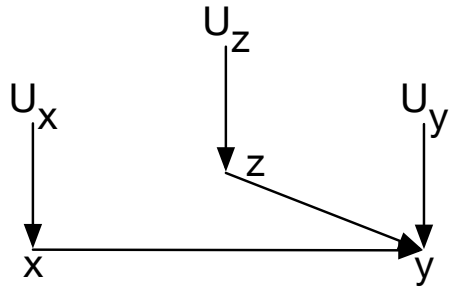
This equation is called the adjustment formula.

Computes the association between X and Y for each z of Z then averages over those values.

The right hand side of the equation can be estimated directly from data, since it consists only of conditional probabilities, each of which can be computed by the filtering procedure! (Pearl's book).

Note: no adjustment is needed in a randomized controlled experiment since, in such a setting, the data are generated by a model which already possesses the structure of the figure on the right and hence, $P = P_m$ regardless of any factors Z that affect Y.

Example



Sets the drug usage in the population, results in the manipulated probability P_m

$$P_m(Y{=}y \mid X{=}x, Z{=}z){=}P(Y{=}y \mid Z{=}z, X{=}x) \text{ and } P_m(Z{=}z){=}P(Z{=}z) \quad (7)$$

Z and X are d-separated in the modified model and are, therefore, independent under the intervention distribution.
This gives $P_m(Z = z \mid X = x) = P_m(Z = z) = P(Z = z)$
So
$P(Y = y \mid do(X = x)) = P_m(Y = y \mid X = x)$ (by definition)
and that leads to Equation (7).

Example

Some derivation:

$$P(Y = y \mid do(X = x)) = P_m(Y = y \mid X = x) \text{ (by definition)}$$
$$= \Sigma_z P_m(Y = y \mid X = x, Z = z) P_m(Z = z \mid X = x)$$
$$= \Sigma_z P_m(Y = y \mid X = x, Z = z) P_m(Z = z)$$

This implies

$$P(Y = y \mid do(X = x)) = \Sigma_z P(Y = y \mid X = x, Z = z) P(Z = z)$$

1. Overview: a brief history of action languages **(20 mnt - Chitta)**
2. Action languages in single agent environments **(70 mnt - Son)**
   - The action language $\mathcal{A}$, state, and transition function
   - $\mathcal{AL}$: $\mathcal{A}+$ static causal laws, non-deterministic and sensing actions
   - GOLOG
   - Action languages: related approaches and planning
3. Action languages and causality **(30 mnt - Chitta)**
   - Pearl's do-calculus
4. Action languages in multi-agent environments **(60 mnt – Son)**
   - mA*, Kripke structure, update models, and transition function
   - mA* in epistemic planning
5. Action languages in commonsense reasoning **(18 mnt - Chitta)**
   - Commonsense reasoning and actions
   - Acquiring knowledge about actions
6. Open challenges, conclusion, and discussion **(12 mnt - Chitta)**

Motivations: from one to many

- Real-world: agents are rarely in isolation
- The presence of multiple agents provides a number of challenges in reasoning about actions and changes

Motivations: from one to many

- Real-world: agents are rarely in isolation
- The presence of multiple agents provides a number of challenges in reasoning about actions and changes
  - there are issues that make sense only when considering in multi-agent environments
    - ⋆ **announcement actions**: someone tells another about a property of the world (lying, misleading, truthful announcement)
    - ⋆ **ontic actions**: the action that changes the world now have additional effects (creating false beliefs for someone); someone executes an action might not know the effects of the action!
  - all types of actions have effects on both knowledge and beliefs of agents implication: needs to deal with both knowledge and beliefs of agents!

Motivations: from one to many

- Real-world: agents are rarely in isolation
- The presence of multiple agents provides a number of challenges in reasoning about actions and changes
  - there are issues that make sense only when considering in multi-agent environments
    - ⋆ **announcement actions**: someone tells another about a property of the world (lying, misleading, truthful announcement)
    - ⋆ **ontic actions**: the action that changes the world now have additional effects (creating false beliefs for someone); someone executes an action might not know the effects of the action!
  - all types of actions have effects on both knowledge and beliefs of agents implication: needs to deal with both knowledge and beliefs of agents!
- **Note**: Philosophers/logician discuss reasoning about knowledge and beliefs of an agent in multi-agent environment for centuries!

  Most earlier frameworks are analogous to transition systems for single-agent environment.

Motivations: from one to many in planning

Single-Agent Planning	**Multi-Agent Planning**
Deliberation process for generating a plan that transforms the state of the world from an initial state to a state satisfying a pre-defined goal	Generalization of the single-agent planning problem to domains where several agents plan and act together and have to share resources, activities, and goals

Motivations

Single-Agent Planning	**Multi-Agent Planning**
Involves generating a plan (sequence of actions, conditional plan, etc.) for the agent to achieve a predefined goal given a problem specification	• involves coordinating the resources and activities of multiple "agents" • is concerned with planning by (and for) multiple agents. It can involve agents planning for a common goal, an agent coordinating the plans (plan merging) or planning of others, or agents refining their own plans while negotiating over tasks or resources.

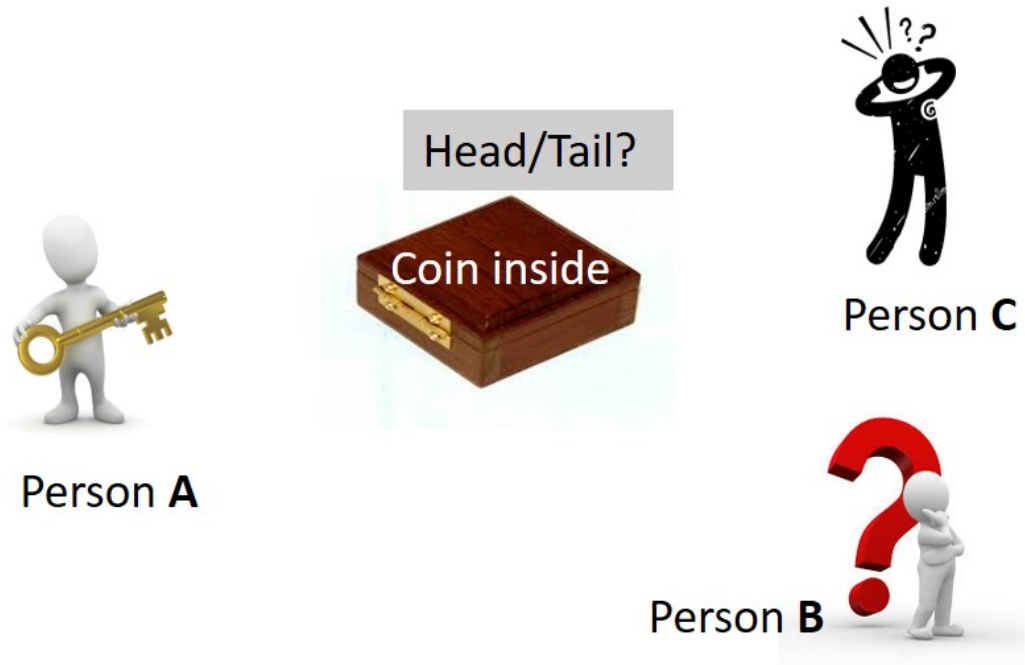
$m\mathcal{A}$: Action Language for Multi-Agent Domains

A Guiding Example

Three agents, A, B, and C, are in a room. In the middle of the room there is a box containing a coin.

- None of the agents knows the state of the coin;
- The box is locked and one needs a key to open it; agent A has the key of the box and everyone knows this;
- To learn whether the coin lies heads or tails up, an agent can peek into the box, if the box is open;
- If one agent is looking at the box and a second agent peeks into the box, then the first agent will conclude that the second agent knows the status of the coin; the first agent's knowledge about the coin does not change;
- Distracting causes that agent to not look at the box;
- Signaling causes such agent to look at the box;
- Announcing the state of the coin will make this a common knowledge among the agents that are listening.

A Guiding Example



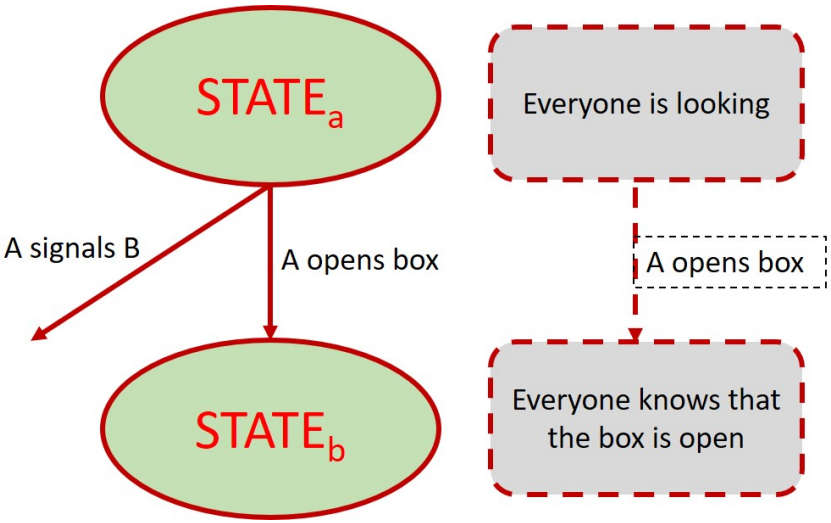
Planning Problem: A knows head/tail and B knows that A knows that the coin lies head/tail up and leaves C in the dark

Solution: A Distracts C, A Signals B, A Opens box, A Looks inside

Challenges: reasoning about knowledge/beliefs of other agents common knowledge—unlimited number of nested knowledge operator

A Guiding Example

- Representation and reasoning
  - Representing beliefs of agents about
    - ★ state of the world
    - ★ state of beliefs of agents
- Defining transition function between states: execution of actions changes
  - state of the world
  - state of beliefs of agents
- Implementation
  - Search algorithm for computing solutions
  - Heuristics



Logic Language

- *Intuition:* Describe properties of the world
  - Fluents **F**

    $has\_key(A)$ $open\_box$
  - Fluent Formulae ($f \in \mathcal{F}$):

$$\psi \ ::= \ \top \mid \bot \mid f \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \psi_1 \Rightarrow \psi_2$$

Logic Language

- *Intuition:* Describe properties of the world
  - Fluents **F**

$$has\_key(A) \qquad open\_box$$

  - Fluent Formulae ($f \in \mathcal{F}$):

$$\psi \ ::= \ \top \mid \bot \mid f \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \psi_1 \Rightarrow \psi_2$$

- *Intuition:* Agents and their Knowledge/Beliefs
  - Agents $\mathcal{AG}$
  - Belief Formulae ($f \in \mathcal{F}$, $a \in \mathcal{AG}$):

$$\varphi \ ::= \ f \mid B_a\varphi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \varphi_1 \vee \varphi_2$$

$$open\_box \wedge B_{bob} heads \ \wedge \ \neg B_{ann} open\_box$$

Logic Language

Brief Comment

- B_a is a modal operator
- Depending on the context, we will read $B_a\varphi$ as

Agent a *knows* φ

or

Agent a *believes* φ

Logic Language

Group Formulae

How to express statements like

"Everyone knows/believes that the box is open"?

- **Group Belief** ($\alpha \subseteq \mathcal{AG}$): $E_\alpha \varphi$
  - $E_{\{bob,ann\}}\, heads$
  - Intuition: $E_\alpha \varphi \equiv \bigwedge_{a \in \alpha} \varphi$

Logic Language

Group Formulae

How to express statements like

"Everyone knows/believes that the box is open"?

- **Group Belief** ($\alpha \subseteq \mathcal{AG}$): $E_\alpha \varphi$
  - $E_{\{bob,ann\}}\, heads$
  - Intuition: $E_\alpha \varphi \equiv \bigwedge_{a \in \alpha} \varphi$
- **Common Belief** ($\alpha \subseteq \mathcal{AG}$): $C_\alpha \varphi$
  - $C_{\{ann,bob,tom\}}\, has\_key(bob)$
  - Intuition: $C_\alpha \varphi \equiv E^*_\alpha \varphi$

Definition

Language $\mathcal{L}(\mathcal{AG}, \mathcal{F})$

Semantics

- Traditional Propositional Logic: *valuation*

$$V : \mathcal{F} \rightarrow \{true, false\}$$

Let $V_{\mathcal{F}}$ set of all valuations over F.

Semantics

- Traditional Propositional Logic: *valuation*

$$V : \mathcal{F} \rightarrow \{true, false\}$$

Let $V_{\mathcal{F}}$ set of all valuations over F.

- *Kripke Structure:* $M = \langle S, \pi, B_{a \in \mathcal{AG}} \rangle$
  - $S \neq \emptyset$ set of worlds (denoted $M[S]$)
  - $\pi : S \rightarrow V_{\mathcal{F}}$ (denoted $M[\pi]$)
  - For each $a \in \mathcal{AG}$: $B_a \subseteq S \times S$ (denoted $M[a]$)

Semantics

- Traditional Propositional Logic: *valuation*

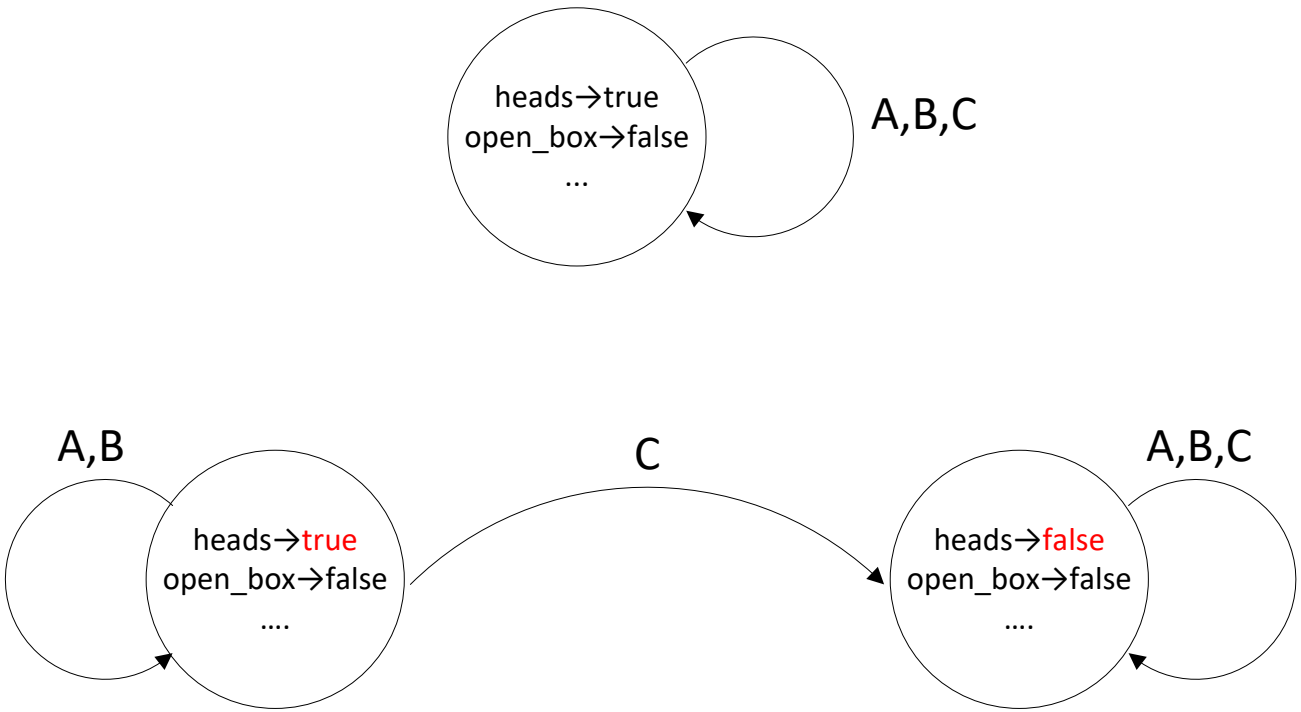
$$V : \mathcal{F} \rightarrow \{true, false\}$$

Let $V_{\mathcal{F}}$ set of all valuations over F.

- *Kripke Structure:* $M = \langle S, \pi, B_{a \in \mathcal{AG}} \rangle$
  - $S \neq \emptyset$ set of worlds (denoted $M[S]$)
  - $\pi : S \rightarrow V_{\mathcal{F}}$ (denoted $M[\pi]$)
  - For each $a \in \mathcal{AG}$: $B_a \subseteq S \times S$ (denoted $M[a]$)
- *State:* (M, s) where
  - M is a Kripke structure
  - $s \in M[S]$ — the "real" state of the world

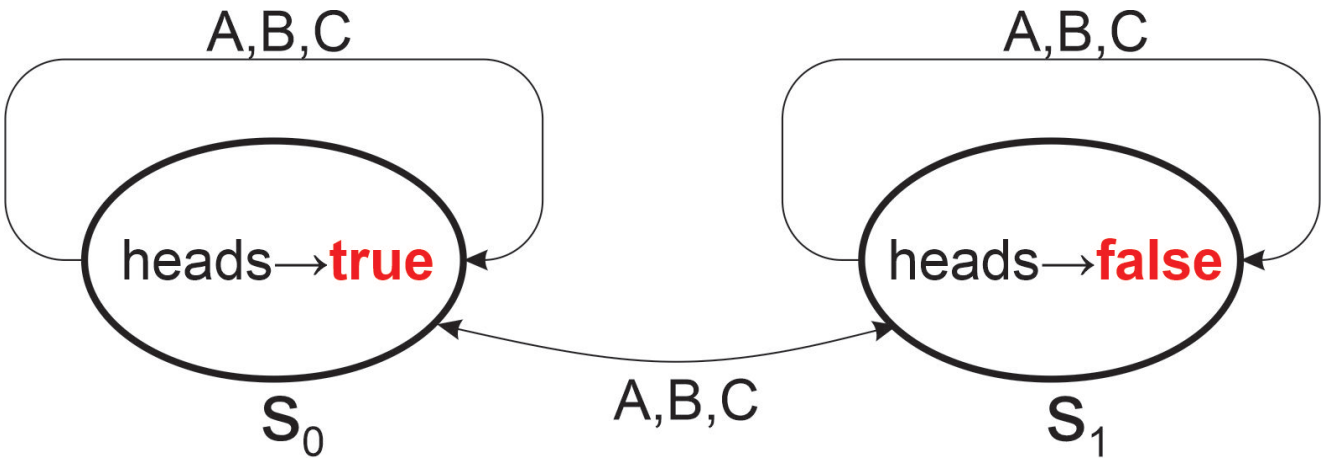
Semantics

Example



Semantics

Example



$$
\begin{array}{lcl}
M[S] & = & \{s_0, s_1\} \\
M[\pi](s_0) & = & \left\{ \begin{array}{ll} heads \rightarrow true, & open \rightarrow false, \\ has\_key(A) \rightarrow true, & has\_key(B) \rightarrow false, \\ has\_key(C) \rightarrow false & \end{array} \right\} \\
M[\pi](s_1) & = & \left\{ \begin{array}{ll} heads \rightarrow false, & open \rightarrow false, \\ has\_key(A) \rightarrow true, & has\_key(B) \rightarrow false, \\ has\_key(C) \rightarrow false & \end{array} \right\} \\
M[A] & = & M[B] = M[C] = \{(s_0, s_0), (s_1, s_1), (s_0, s_1), (s_1, s_0)\}
\end{array}
$$

Semantics

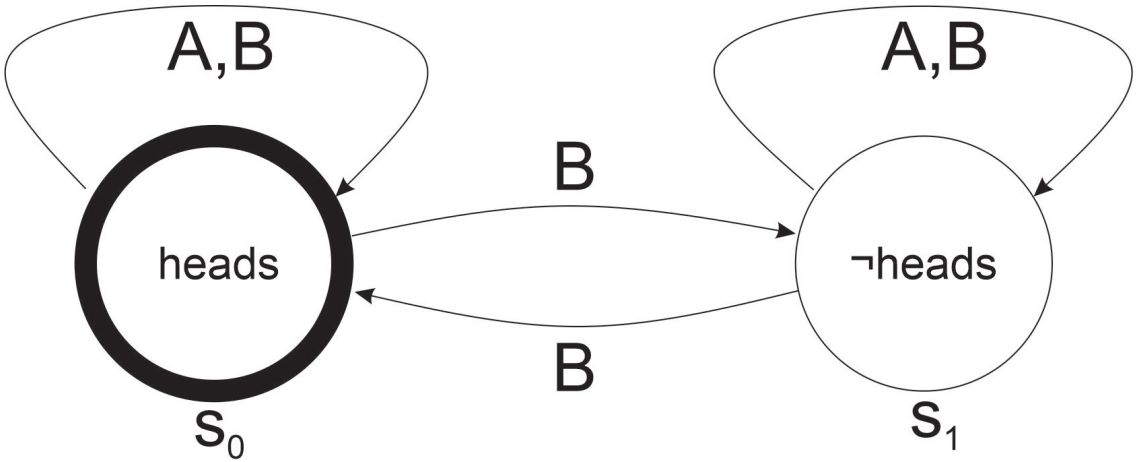
Entailment

- $(M, s) \models f$ iff $M[\pi](s)(f) = \mathit{true}$
- $(M, s) \models \varphi_1 \wedge \varphi_2$ iff $(M, s) \models \varphi_1$ and $(M, s) \models \varphi_2$
- $(M, s) \models \neg\varphi$ iff $(M, s) \not\models \varphi$
- $(M, s) \models B_a\varphi$ iff for all t such that $(s, t) \in M[a]$ we have $(M, t) \models \varphi$

Semantics

Entailment

- $(M, s) \models f$ iff $M[\pi](s)(f) = true$
- $(M, s) \models \varphi_1 \wedge \varphi_2$ iff $(M, s) \models \varphi_1$ and $(M, s) \models \varphi_2$
- $(M, s) \models \neg\varphi$ iff $(M, s) \not\models \varphi$
- $(M, s) \models B_a\varphi$ iff for all t such that $(s, t) \in M[a]$ we have $(M, t) \models \varphi$



$(M, s_0) \models heads$
$(M, s_1) \models \neg heads$
$(M, s_0) \models B_A heads$
$(M, s_0) \not\models B_B heads$

Semantics: Entailment

Group Formulae

- $(M, s) \models E_\alpha \varphi$ iff $\forall a \in \alpha$ we have $(M, s) \models B_a \varphi$
- $(M, s) \models C_\alpha \varphi$ iff $(M, s) \models E_\alpha^k \varphi$ for $k \geq 0$
  - $E_\alpha^0 \varphi \equiv \varphi$
  - $E_\alpha^{k+1} \equiv E_\alpha(E_\alpha^k \varphi)$

Semantics: Axioms

- **K:** $(B_a\varphi_1 \wedge B_a(\varphi_1 \Rightarrow \varphi_2)) \Rightarrow B_a\varphi_2$
- **D:** $\neg B_a\bot$
- **4:** $B_a\varphi \Rightarrow B_aB_a\varphi$
- **5:** $\neg B_a\varphi \Rightarrow B_a\neg B_a\varphi$
- **T:** $B_a\varphi \Rightarrow \varphi$

Semantics: Axioms

- **K:** $(B_a\varphi_1 \wedge B_a(\varphi_1 \Rightarrow \varphi_2)) \Rightarrow B_a\varphi_2$
- **D:** $\neg B_a\bot$
- **4:** $B_a\varphi \Rightarrow B_aB_a\varphi$
- **5:** $\neg B_a\varphi \Rightarrow B_a\neg B_a\varphi$
- **T:** $B_a\varphi \Rightarrow \varphi$

Some typical axiomatic systems:

- **KD45:** Considered typical modeling of *Beliefs*
- **KT45 (S5):** Modeling of *Knowledge*

Semantics: Axioms

Implications on Kripke Structure M:

- **KD45:** All accessibility relations $M[a]$ are
  - *Serial:* for each $s \in M[S]$ there is a t such that $(s, t) \in M[a]$
  - *Transitive*
  - *Euclidean:* for all $(s, t) \in M[a]$ and $(s, u) \in M[a]$ then $(t, u) \in M[a]$

- **S5:** all accessiblity relations $M[a]$ are
  - *Reflexive*
  - *Symmetric*
  - *Transitive*

Action Models

A Logic of Communication and Change

$\mathcal{L}(\mathcal{AG}, \mathcal{F})$-Substitution:

$$\{f \mapsto \varphi \mid f \in \mathcal{F}, \varphi \in \mathcal{L}(\mathcal{AG}, \mathcal{F})\}$$

$SUB_{\mathcal{L}}$ set of all $\mathcal{L}(\mathcal{AG}, \mathcal{F})$-Substitutions

An action occurrence could be perceived as different event by different agent.

Action Models

A Logic of Communication and Change

$\mathcal{L}(\mathcal{AG}, \mathcal{F})$-Substitution:

$$\{f \mapsto \varphi \mid f \in \mathcal{F}, \varphi \in \mathcal{L}(\mathcal{AG}, \mathcal{F})\}$$

$SUB_{\mathcal{L}}$ set of all $\mathcal{L}(\mathcal{AG}, \mathcal{F})$-Substitutions

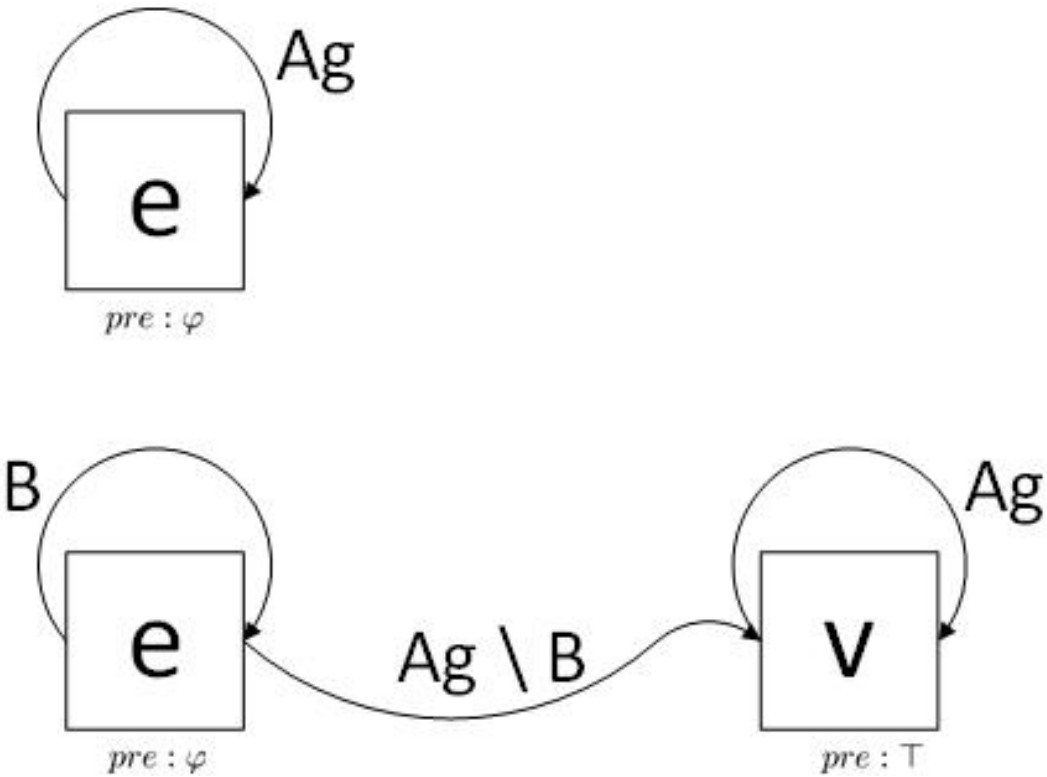
Action Model: $\mathbf{\Sigma} = \langle \Sigma, R_{a \in \mathcal{AG}}, pre, sub \rangle$ where

- Σ is a set of *events*
- for each $a \in \mathcal{AG}$ we have that $R_a \subseteq \Sigma \times \Sigma$
- $pre : \Sigma \to \mathcal{L}(\mathcal{AG}, \mathcal{F})$ (preconditions)
- $sub : \Sigma \to SUB_{\mathcal{L}}$ (substitutions)

Action Instance: $(\mathbf{\Sigma}, e)$ with $e \in \Sigma$

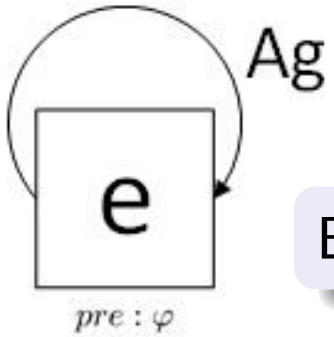
Action Models

Examples

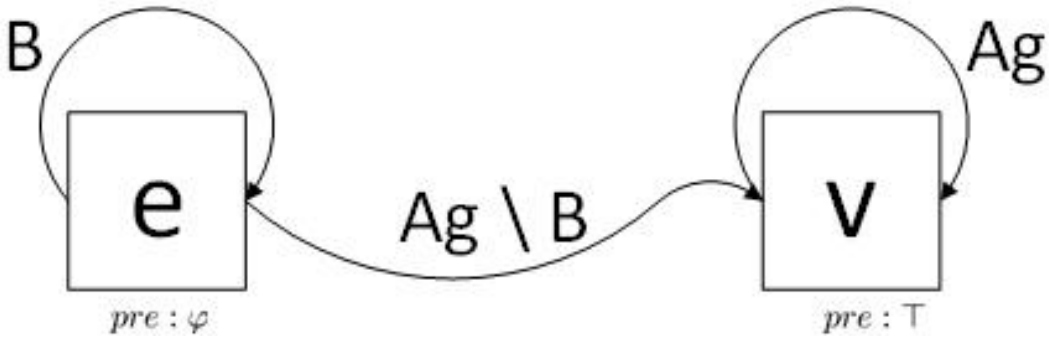


Action Models

Examples



Everyone knows that e happens.



B knows that e happens and everyone else knows that v happens.

Updates

Given

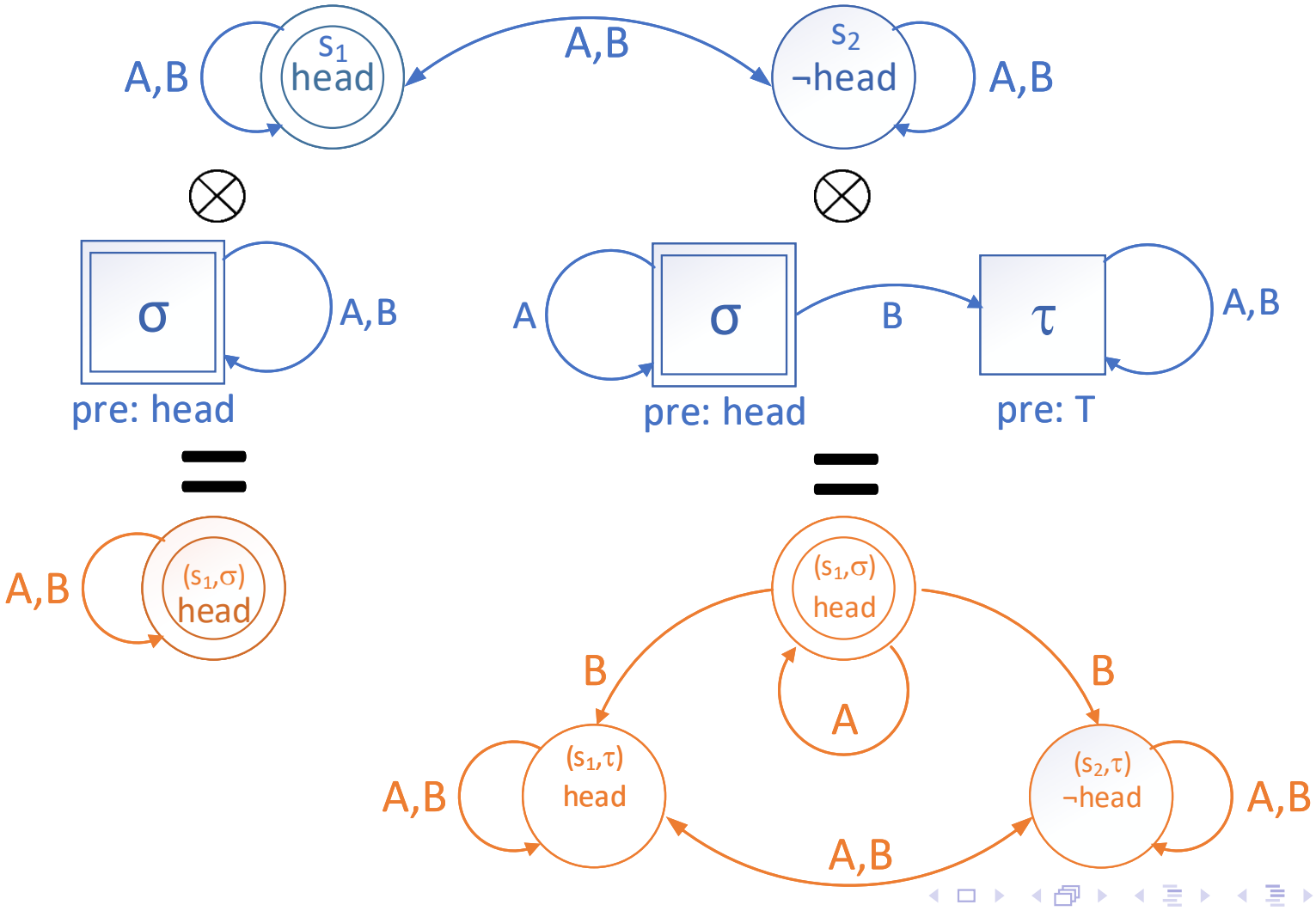
- $M = \langle S, \pi, B_{a \in \mathcal{AG}} \rangle$ Kripke Structure
- $\mathbf{\Sigma} = \langle \Sigma, R_{a \in \mathcal{AG}}, pre, sub \rangle$ Action Model

Update of M by $\mathbf{\Sigma}$, $M' = M \otimes \mathbf{\Sigma}$:

- $M'[S] = \{(s, e) \mid s \in M[S], e \in \Sigma, (M, s) \models pre(e)\}$
- $((s_0, e_0), (s_1, e_1)) \in M'[a]$ iff
  - $(s_0, e_0) \in M'[S]$ and $(s_1, e_1) \in M'[S]$
  - $(s_0, s_1) \in M[a]$
  - $(e_0, e_1) \in R_a$
- $M'[\pi](s, e)(f) = true$ iff $f \mapsto \varphi$ in $sub(e)$ and $(M, s) \models \varphi$

Updates

Example



The Language $m\mathcal{A}^*$: Why?

- Kripke structure is suitable for reasoning about knowledge and beliefs of multi-agents.
- Action model could be used for representing and reasoning about actions and changes

The Language $m\mathcal{A}^*$: Syntax

Motivations

Back to the Example

Suppose that the agent **A** would like to know whether the coin lies heads or tails up. She would also like to let the agent **B** know that she knows this fact. However, she would like to keep this information secret from **C**.

1. Distract **C** from looking at the box;
2. Signal **B** to look at the box if **B** is not looking at the box;
3. Open the box; and
4. Peek into the box.

The Language $m\mathcal{A}^*$: Syntax

Motivations

Challenges:

- Solid logical foundations—model-theoretic, not amenable to implementation in a search-based planner
- DEL:
  - Action Models are really more like action occurrences
  - May require infinitely many conditional effects to account for all possible perceptions of the action
  - Even when finite, Action Models may have to be big
    - ★ Extreme case: **B** has no idea about whether **A** performs `open` or `peek`; nor what **A** knows about **B**'s perception of this; nor... $\Rightarrow$ action model is infinite

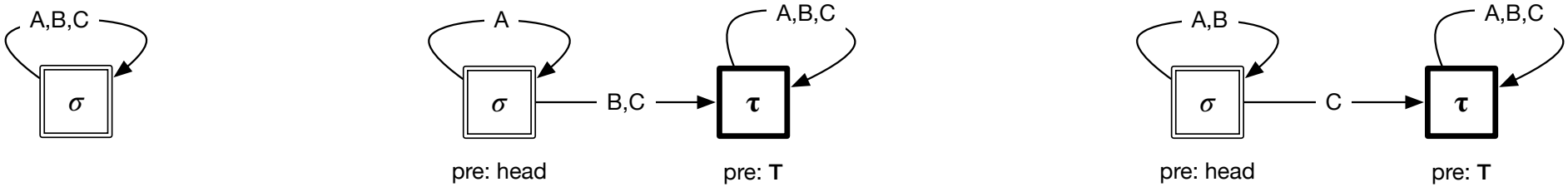
Goal

Knowledge Representation = Representation + Reasoning

The Language $m\mathcal{A}^*$: Syntax

Motivations

A has looked at the coin, while both **B** and **C** are distracted, and **A** can announce whether the coin lies heads up. However, only agents who are attentive could listen to what **A** says. Thus, the action occurrence can have different effects on the beliefs of the other agents—e.g., whether the agent is attentive to **A**.



$m\mathcal{A}^*$: Syntax

Basics

Language components:

- $\mathcal{AG}$ set of agent names—e.g., **A**, **B**, ...
- F set of fluents—e.g., *heads*, *has_key*(**A**), *looking*(**B**)
- A set of actions—e.g., *open*, *signal*(**C**)

The Language $m\mathcal{A}^*$: Syntax

Actions

Action Instance

$$\underbrace{a}_{a \in A} \underbrace{\langle \alpha \rangle}_{\alpha \subseteq \mathcal{AG}}$$

Intuition: action a jointly executed by agents α

Executability Conditions

executable $\underbrace{a\langle \alpha \rangle}_{\substack{\text{action} \\ \text{instance}}}$ **if** φ

Example: **executable** $open\langle x \rangle$ **if** $has\_key(x)$

executable $signal(y)\langle x \rangle$ **if** $looking(x) \wedge \neg looking(y)$

Actions

Action Types:

- *World Altering (ontic):* $a\langle\alpha\rangle$ **causes** ℓ **if** φ

$$signal(y)\langle x\rangle \textbf{ causes } looking(y)$$

- *Sensing:* $a\langle\alpha\rangle$ **determines** f

$$peek\langle x\rangle \textbf{ determines } heads$$

- *Annoucement:* $a\langle\alpha\rangle$ **announces** φ

$$shout\_head\langle x\rangle \textbf{ announces } head$$

Comparing to $\dashv$

- The set of agents is attached.
- Announcement action.

The Language $m\mathcal{A}^*$: Syntax

Visibility

Agent A

Fully Observant

Agent B

Partial Observer

Agent C

Oblivious

Syntax

Visibility

Visibility of Action Effects:

- *Full Observers:* x **observes** $a\langle\alpha\rangle$ **if** φ

  y **observes** $open\langle x\rangle$ **if** $looking(y)$

- *Partial Observers:* x **aware_of** $a\langle\alpha\rangle$ **if** φ

  y **aware_of** $peek\langle x\rangle$ **if** $looking(y)$

- *Oblivious*

Syntax

Visibility

Action Type	Full Observers	Partial Observes	Oblivious
Ontic	√		√
Sensing	√	√	√
Announcement	√	√	√

Syntax

Domain

Domain

A $m\mathcal{A}^*$ domain is a collection of executability statements, action descriptions, and visibility statements.

Assumption: action domains are consistent, i.e., for each pairs of action descriptions

$$\textbf{a causes } f \textbf{ if } \varphi \qquad\qquad \textbf{a causes } \neg f \textbf{ if } \psi$$

and each state (M, s) we have that $(M, s) \not\models \varphi \wedge \psi$.

Action Theory

A $m\mathcal{A}^*$ theory is a pair (D, I) where D is a $m\mathcal{A}^*$ domain and I is a collection of statements of the type

$$\textbf{initially } \varphi$$

Syntax

Example

initially $looking(A)$
initially $C_{\{A,B,C\}}(looking(A))$
initially $C_{\{A,B,C\}}(has\_key(A))$
initially $C_{\{A,B,C\}}(\neg B_A heads \ \wedge \ \neg B_A \neg heads)$

The Language $m\mathcal{A}^*$: Transition Function

Intuition: Action language semantics typically based on a *transition function*:

$$\Phi : Action \times State \rightarrow 2^{State}$$

Our goal: $\Phi_D : AI \times S \rightarrow 2^S$ where:

- AI: set of all action instances
- S: set of all states (i.e., (M, s))

The Language $m\mathcal{A}^*$: Transition Function

Preliminary Definitions

Frames of Reference: Given state (M, s) and action instance $\mathbf{a}$:

$Full(\mathbf{a}, M, s) = \{x \in \mathcal{AG} \mid [x \textbf{ observes a if } \varphi], (M, s) \models \varphi\}$

$Part(\mathbf{a}, M, s) = \{x \in \mathcal{AG} \mid [x \textbf{ aware\_of a if } \varphi], (M, s) \models \varphi\}$

$Obl(\mathbf{a}, M, s) = \mathcal{AG} \setminus (Full(\mathbf{a}, M, s) \cup Part(\mathbf{a}, M, s))$

Note: actions change the frame of reference of future actions;

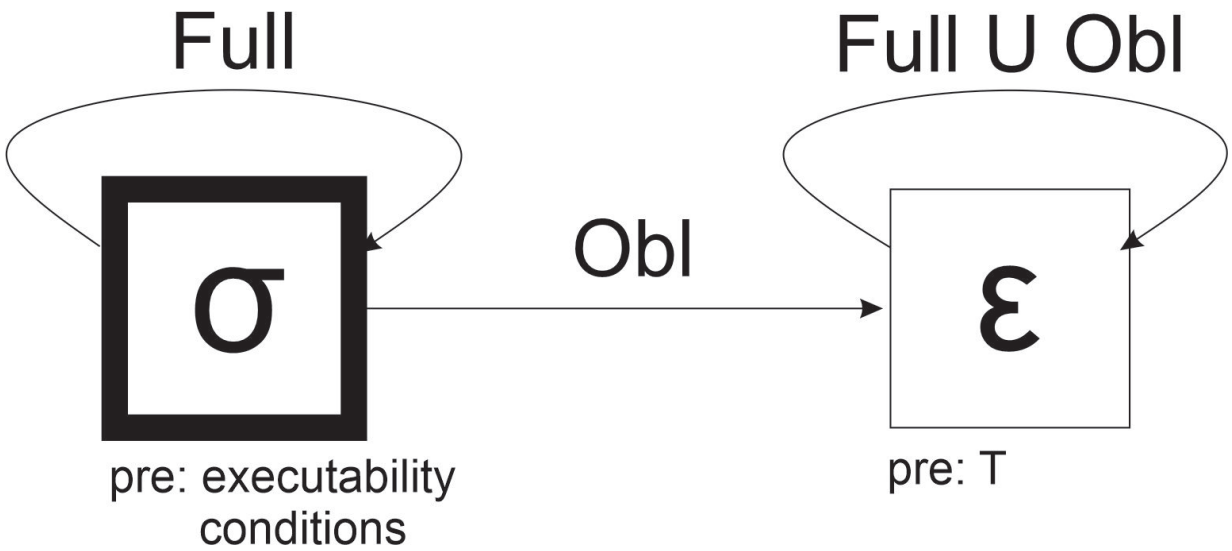
- $\mathbf{a} = peek\langle A \rangle$
- (M, s)
- execution of $signal(B)\langle A \rangle$ in (M, s) produces state (M', s') where
  - $Full(\mathbf{a}, M', s') = Full(\mathbf{a}, M, s)$
  - $Part(\mathbf{a}, M', s') = Part(\mathbf{a}, M, s) \cup \{B\}$
  - $Obl(\mathbf{a}, M', s') = Obl(\mathbf{a}, M, s) \setminus \{B\}$

The Language $m\mathcal{A}^*$: Transition Function

Action Models

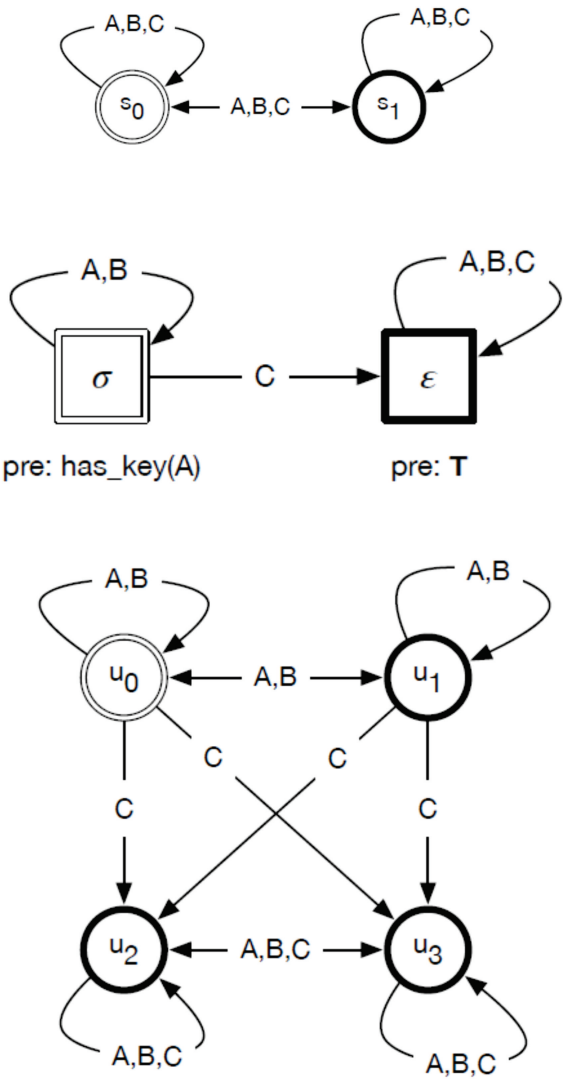
Ontic Actions: Intuition: two possible events

1. (σ) The action is seen by the agents
   1. substitution needs to reflect the effects of the action
   2. for all agents in $Full(\mathbf{a}, M, s)$
2. (ϵ) The agents are unaware of the action
   1. substitutions make no change
   2. for all agents in $Obl(\mathbf{a}, M, s)$



The Language $m\mathcal{A}^*$: Transition Functions

Example Ontic



Transition Function

Sensing Actions

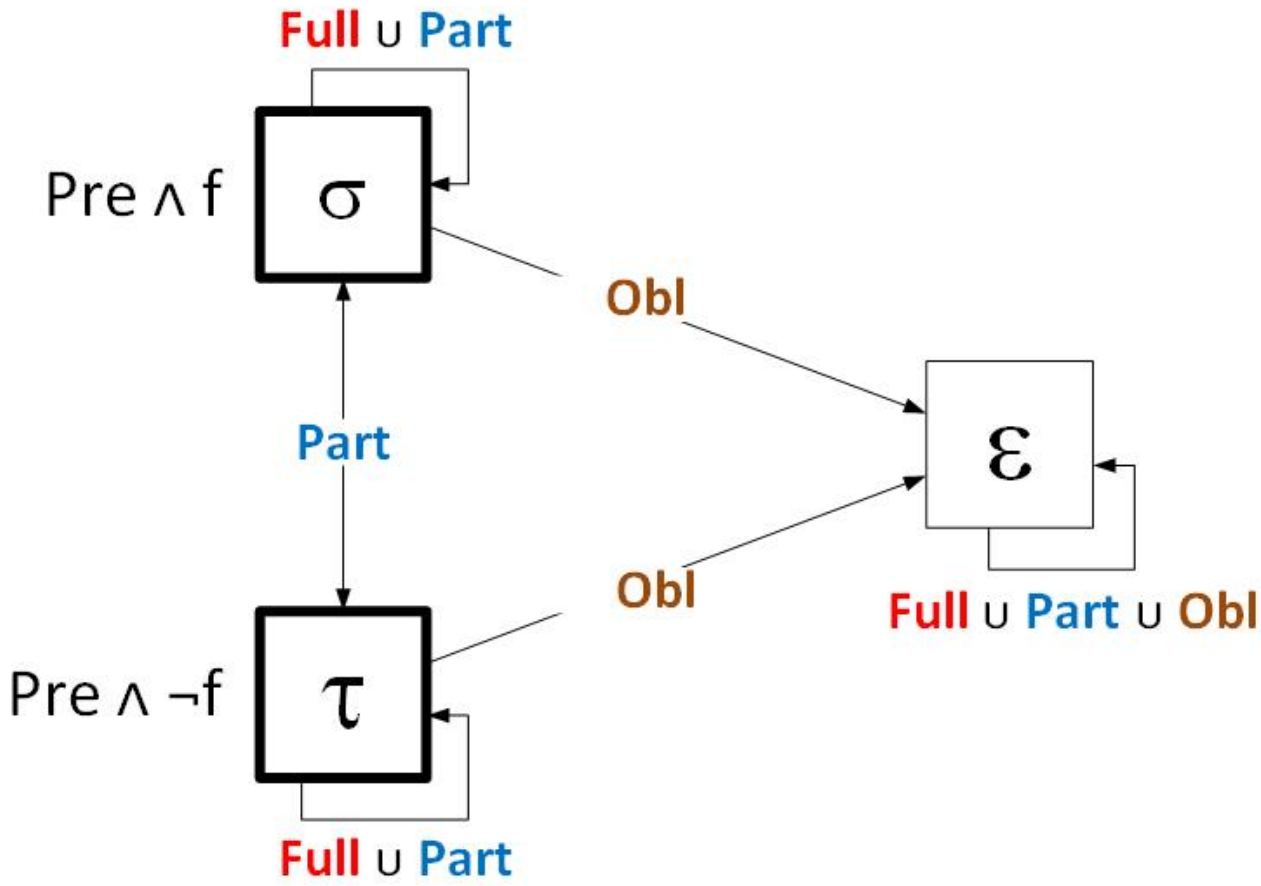
Sensing Actions: Intuition: sensing fluent f, three possible events

1. (σ) the action is seen and f is true
   1. For all agents in $Full(\mathbf{a}, M, s)$
2. (τ) the action is seen and f is false
   1. For all agents in $Full(\mathbf{a}, M, s)$
3. $(\sigma$ and $\tau)$
   1. For all agents in $Part(\mathbf{a}, M, s)$
4. (ϵ) the agents are unaware of the action
   1. For all agents in $Obl(\mathbf{a}, M, s)$

Since no change of world, all substitutions are empty.

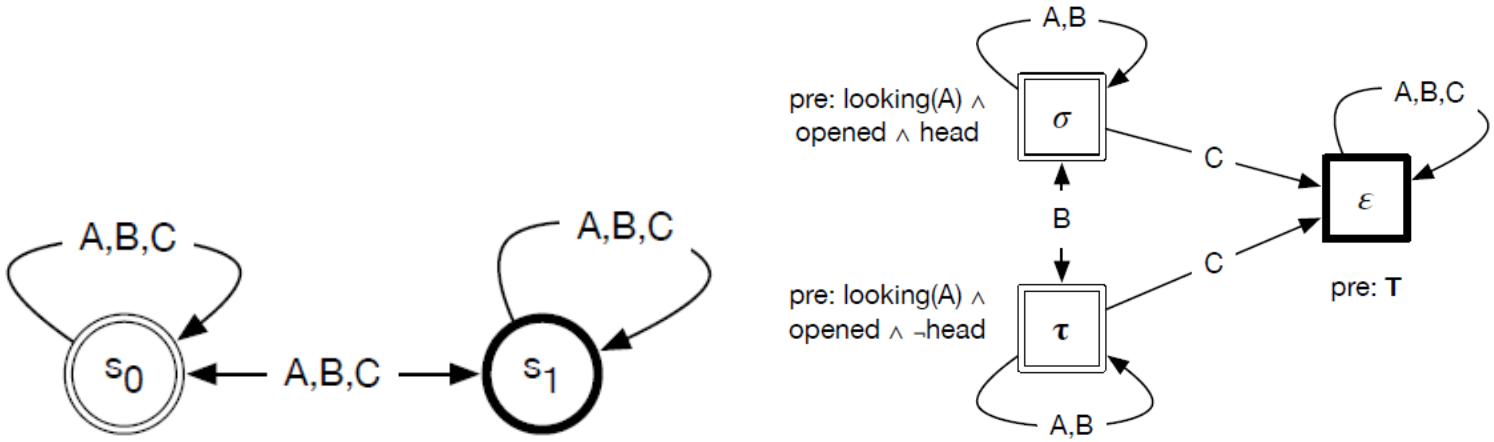
The Language $m\mathcal{A}^*$: Transition Function

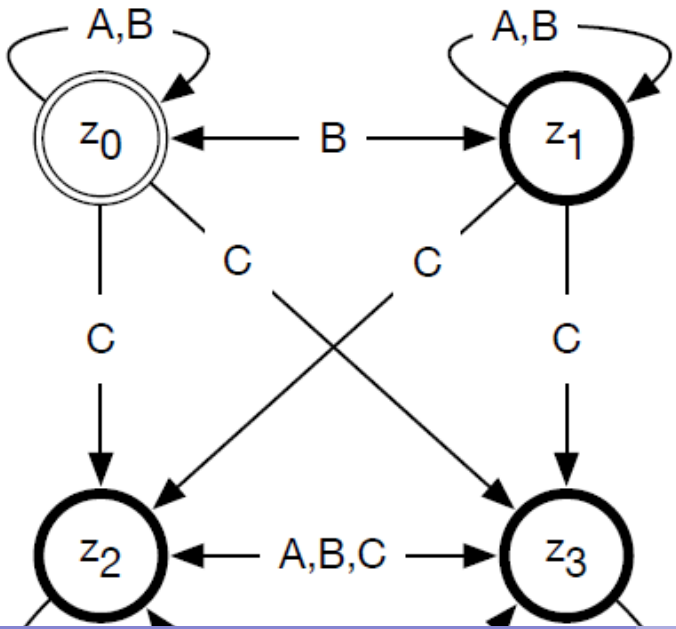
Sensing Actions



Transition Function

Example Sensing: peek(A)





The Language $m\mathcal{A}^*$: Transition Function

Announcement Actions

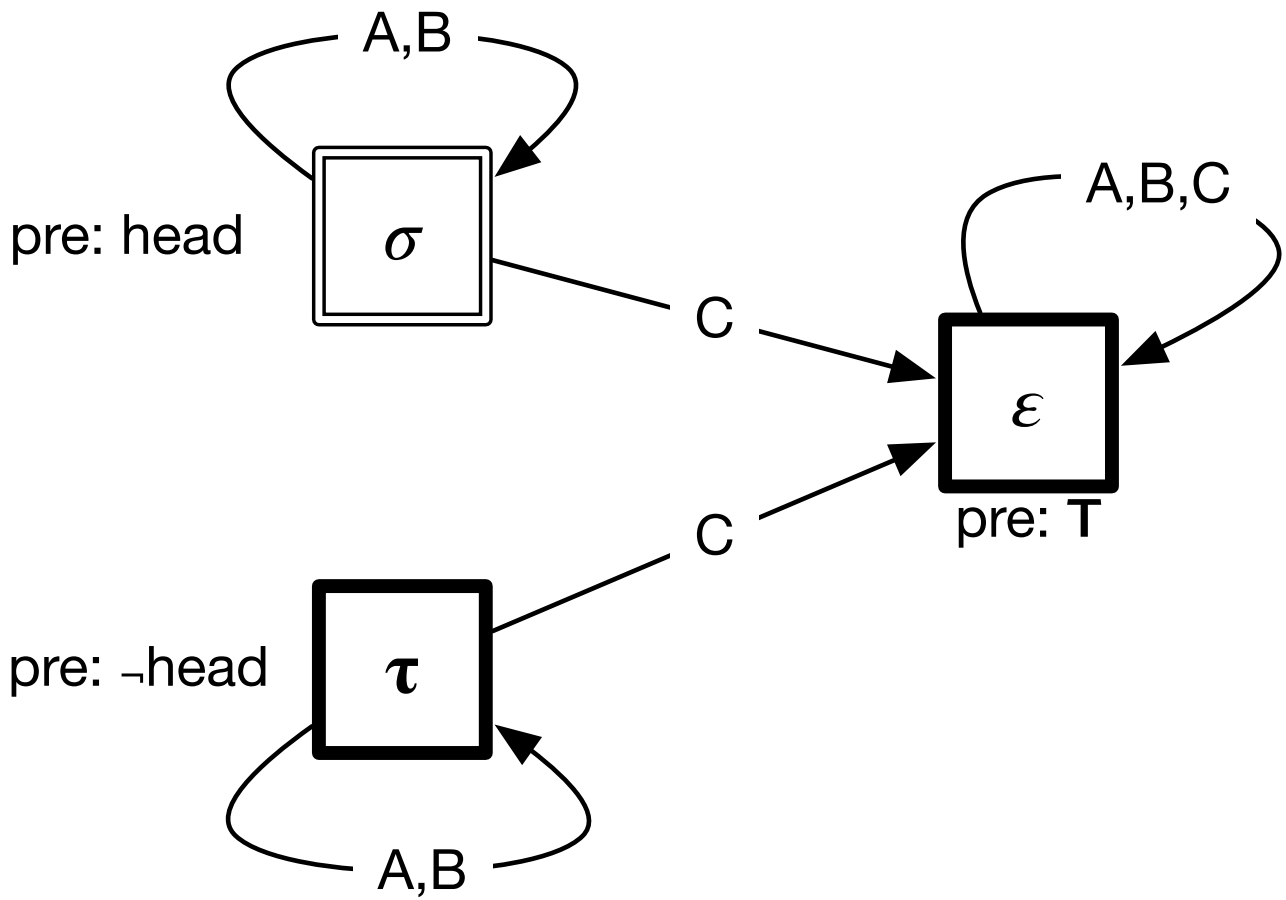
Announcement Actions: Intuition: announcing formula φ, three possible events

1. (σ) the action is seen and φ is true
   1. For all agents in $Full(\mathbf{a}, M, s)$
2. (τ) the action is seen and φ is false
   1. For all agents in $Full(\mathbf{a}, M, s)$
3. $(\sigma$ and $\tau)$
   1. For all agents in $Part(\mathbf{a}, M, s)$
4. (ϵ) the agents are unaware of the action
   1. For all agents in $Obl(\mathbf{a}, M, s)$

Since no change of world, all substitutions are empty.

The Language $m\mathcal{A}^*$: Transition Function

Example Announcement: whisper_head



The Language $m\mathcal{A}^*$: Transition Function

Let $(\mathcal{E}, E_d)$ be the action models for action occurrence **a** in (M, s).

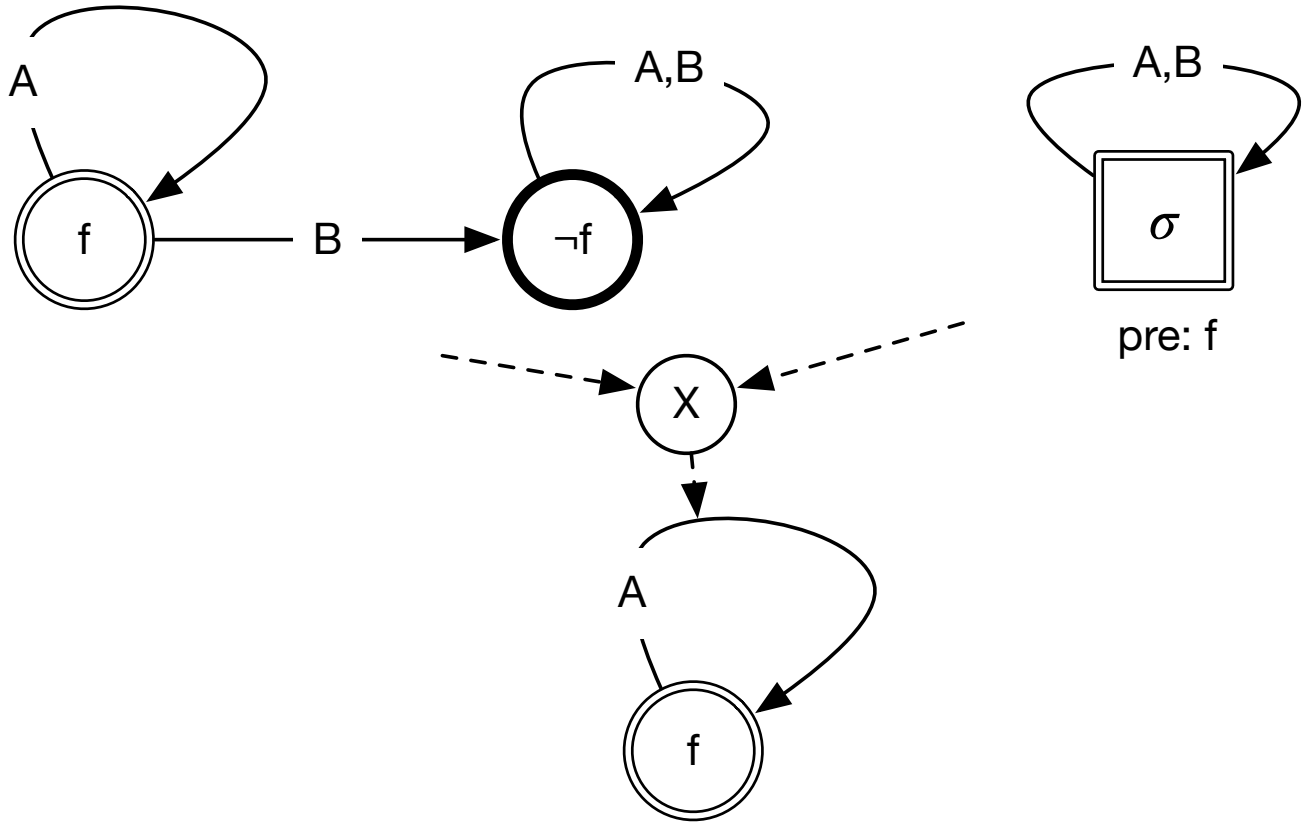
Temptation: Define

$$\Phi_D(\mathbf{a}, M, s) = \bigcup_{e \in E_d} (M, s) \otimes (\mathcal{E}, e)$$

But...

Transition Function

Problem: False Beliefs

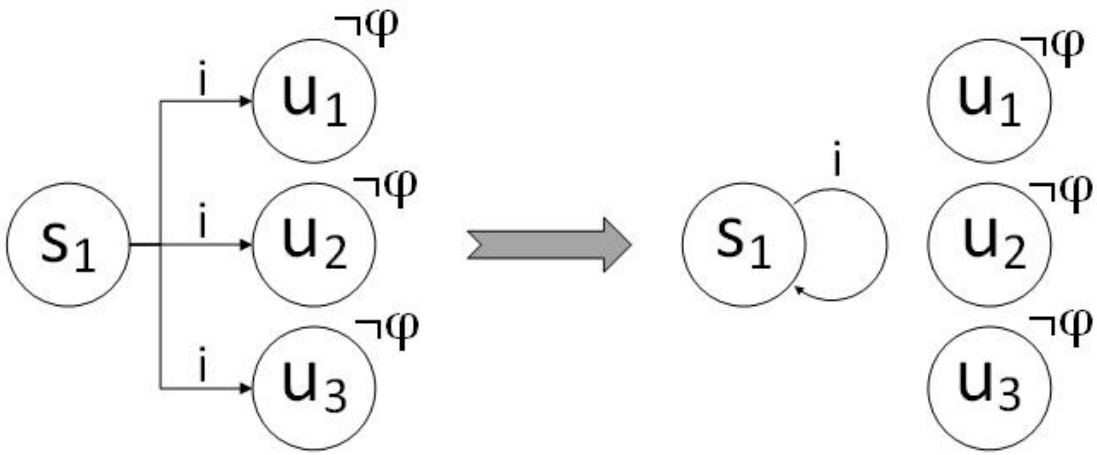


The Language $m\mathcal{A}^*$: Transition Function

Challenge: False beliefs

$$(M, s) \models \phi \textit{ but } (M, s) \models B_i \neg\varphi$$

Repair: Given a Kripke structure M and a set of agents S,
Repair(M, S, φ): for each $i \in S$ and s_1 such that $(M, s_1) \models B_i \neg\varphi$



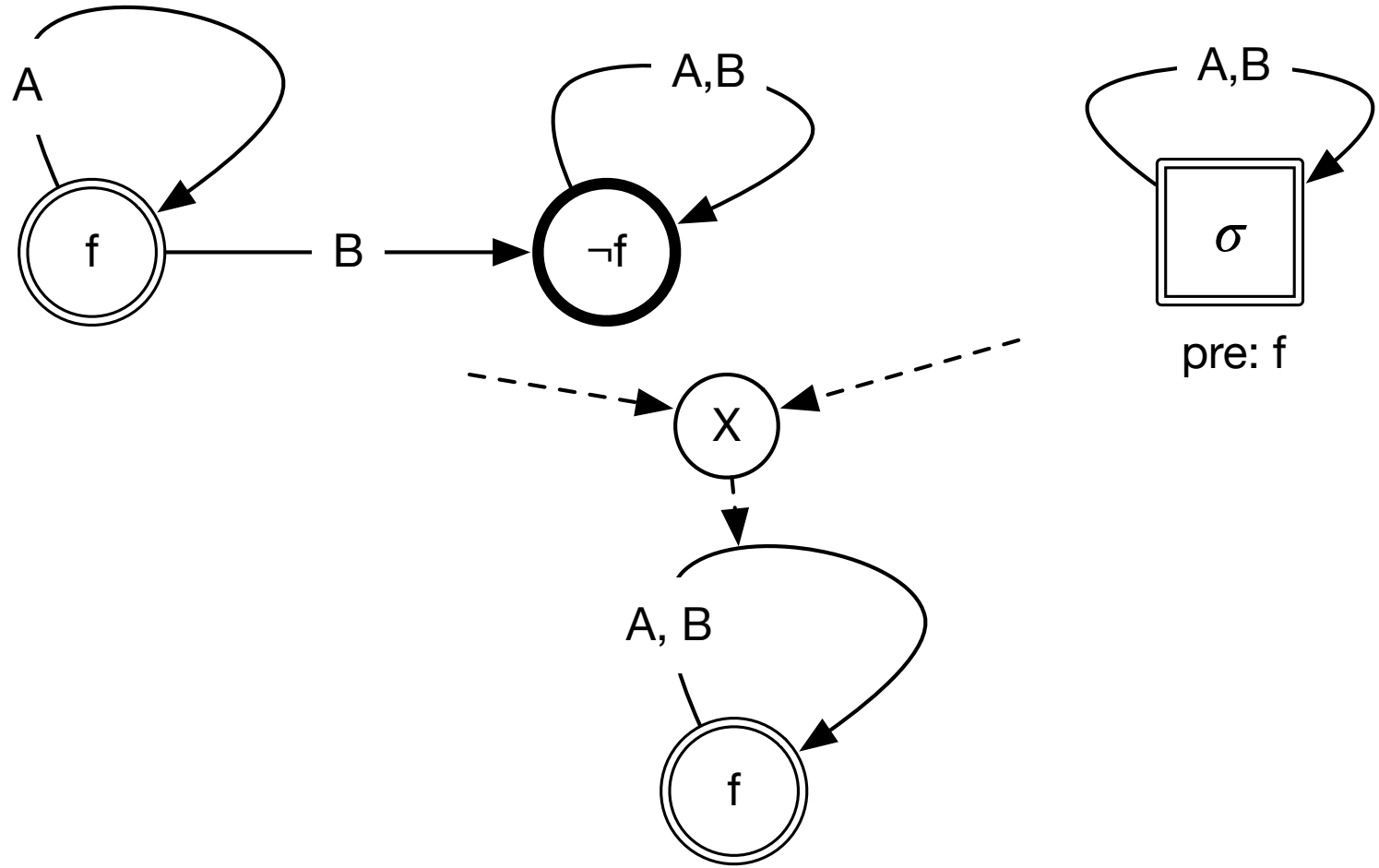
The Language $m\mathcal{A}^*$: Transition Function

Execute action instance $\mathbf{a} = a\langle\alpha\rangle$ in (M, s) with action model (E, E_D)

1. action executable in (M, s) if preconditions are satisfied
2. For ontic actions $\Phi_D(\mathbf{a}, M, s) = (M, s) \otimes (E, E_D)$
3. For sensing actions where $(M, s) \models f$ and f is sensed
   $\Phi_D(\mathbf{a}, M, s) = \mathit{Repair}(M, \mathit{Full}(\mathbf{a}, M, s), f) \otimes (E, E_D)$
4. For sensing actions where $(M, s) \models \neg f$ and f is sensed
   $\Phi_D(\mathbf{a}, M, s) = \mathit{Repair}(M, \mathit{Full}(\mathbf{a}, M, s), \neg f) \otimes (E, E_D)$
5. for announcement action for φ then
   $\Phi_D(\mathbf{a}, M, s) = \mathit{Repair}(M, \mathit{Full}(\mathbf{a}, M, s), \varphi) \otimes (E, E_D)$

The Language $m\mathcal{A}^*$: Transition Function

Example



Action Properties: Ontic Actions

a causes ℓ if φ

- if $x \in Full(\mathbf{a}, M, s)$ and $(M, s) \models B_x\varphi$ then $\Phi_D(\mathbf{a}, M, s) \models B_x\ell$
- if $x \in Obl(\mathbf{a}, M, s)$ and $(M, s) \models B_x\eta$ then $\Phi_D(\mathbf{a}, M, s) \models B_x\eta$
- if $x \in Full(\mathbf{a}, M, s)$ and $y \in Obl(\mathbf{a}, M, s)$ and $(M, s) \models B_xB_y\eta$ then $\Phi_D(\mathbf{a}, M, s) \models B_xB_y\eta$

Action Properties: Sensing Actions

a determines f

- if $(M, s) \models f$ then $\Phi_D(\mathbf{a}, M, s) \models C_{Full(\mathbf{a},M,s)} f$
- if $(M, s) \models \neg f$ then $\Phi_D(\mathbf{a}, M, s) \models C_{Full(\mathbf{a},M,s)} \neg f$
- $\Phi_D(\mathbf{a}, M, s) \models C_{Part(\mathbf{a},M,s)}(C_{Full(\mathbf{a},M,s)} f \vee C_{Full(\mathbf{a},M,s)} \neg f)$
- if $x \in Obl(\mathbf{a}, M, s)$ and $(M, s) \models B_x \eta$ then $\Phi_D(\mathbf{a}, M, s) \models B_x \eta$
- if $x \in Full(\mathbf{a}, M, s)$ and $y \in Obl(\mathbf{a}, M, s)$ and $(M, s) \models B_x B_y \eta$ then $\Phi_D(\mathbf{a}, M, s) \models B_x B_y \eta$

Action Properties: Announcement Actions

a announces φ and $(M, s) \models \varphi$

- $\Phi_D(\mathbf{a}, M, s) \models C_{Full(\mathbf{a},M,s)}\varphi$
- $\Phi_D(\mathbf{a}, M, s) \models C_{Part(\mathbf{a},M,s)}(C_{Full(\mathbf{a},M,s)}\varphi \vee C_{Full(\mathbf{a},M,s)}\neg\varphi)$
- if $x \in Obl(\mathbf{a}, M, s)$ and $(M, s) \models B_x\eta$ then $\Phi_D(\mathbf{a}, M, s) \models B_x\eta$
- if $x \in Full(\mathbf{a}, M, s)$ and $y \in Obl(\mathbf{a}, M, s)$ and $(M, s) \models B_xB_y\eta$ then $\Phi_D(\mathbf{a}, M, s) \models B_xB_y\eta$

Initial State

Initial state described by a collection I of statements:

initially φ

Single agent domain: If the set of propositions is finite then a theory has only finitely many finite models.
Existing literature: a single initial Kripke state
Multi-agent domains:

- Models of a theory can be infinite.
- If a theory is consistent (has a model) then it has a finite model.
- Adding common knowledge operator **C** usually increases complexity.
- In multi-modal logics, a theory can have infinitely many infinite models even for finite set of propositions.

Initial State

Assumption

- Initial state focuses on what is known
- Limit attention to **S5** states

Some preliminary definitions:

- (M, s) model of theory T if $(M, s) \models \psi$ for each $\psi \in T$
- (M, s) equivalent to (M', s'): for each $\psi \in \mathcal{L}(\mathcal{AG}, \mathcal{F})$

$$(M, s) \models \psi \quad \textit{iff} \quad (M', s') \models \psi$$

Initial State

More preliminary definitions:

- A *complete clause* over F is a disjunction of literals $\bigvee_{i=1}^{|\mathcal{F}|} \ell_i$ where $\{f_i \mid \ell_i = f_i \vee \ell_i = \neg f_i\} = \mathcal{F}$

Primitive Formulae

1. φ fluent formula
2. $C(B_i\varphi)$ where φ fluent formula
3. $C(B_i\varphi \vee B_i\neg\varphi)$ where φ fluent formula
4. $C(\neg B_i\varphi \wedge \neg B_i\neg\varphi)$ where φ fluent formula

Note:

- we write $K_i\varphi$ instead of $B_i\varphi$
- note that $C(K_i\varphi)$ is equivalent to $C(\varphi)$

Initial State

Primitive Finitary S5-Theory T

- T is composed of primitive formulae
- For each complete clause ψ of F and agent i, T contains:
  - $C(K_i\psi)$, or
  - $C(K_i\psi \vee K_i\neg\psi)$, or
  - $C(\neg K_i\psi \wedge \neg K_i\neg\psi)$

Finitary S5-Theory T

T is a Finitary S5 Theory if there exists a Primitive Finitary S5 theory H such that $T \models H$.

Initial State

Theorem

For a consistent primitive finitary S5-theory T there is a finite set $\mu ModsS5(T)$ such that

1. *Each model in $\mu ModsS5(T)$ is finite;*
2. *Each S5 model (M, s) of T is equivalent to one model in $\mu ModsS5(T)$.*

Theorem

Every Finitary S5 Theory T has finitely many finite canonical[a] models, up to equivalence.

[a] A model (M, s) is canonical if for each $u, v \in M[S]$ $M[\pi](u) \neq M[\pi](v)$.

Initial State

Example

1. φ
   1. **initially** $has\_key(A)$
   2. **initially** $heads$
2. $C(K_i\varphi)$
   1. **initially** $C(K_A has\_key(A))$
3. $C(K_i\varphi \vee K_i\neg\varphi)$
   1. **initially** $C(K_A heads \vee K_A\neg heads)$
4. $C(\neg K_i\varphi \wedge \neg K_i\neg\varphi)$
   1. **initially** $C(\neg K_B heads \wedge \neg K_B\neg heads)$

Computing Initial States: [Son et al. (2014)]

Forward Search Planner [Le et al. (2018)]

EPF and PG-EPF

Components of a forward search planner:

- *Pre-Processor:* Parsing and build data structure
- *Initial States Computation*
- *Search Module*
  - *EPF:* breadth-first search
  - *PG-EPF:* heuristic search planner

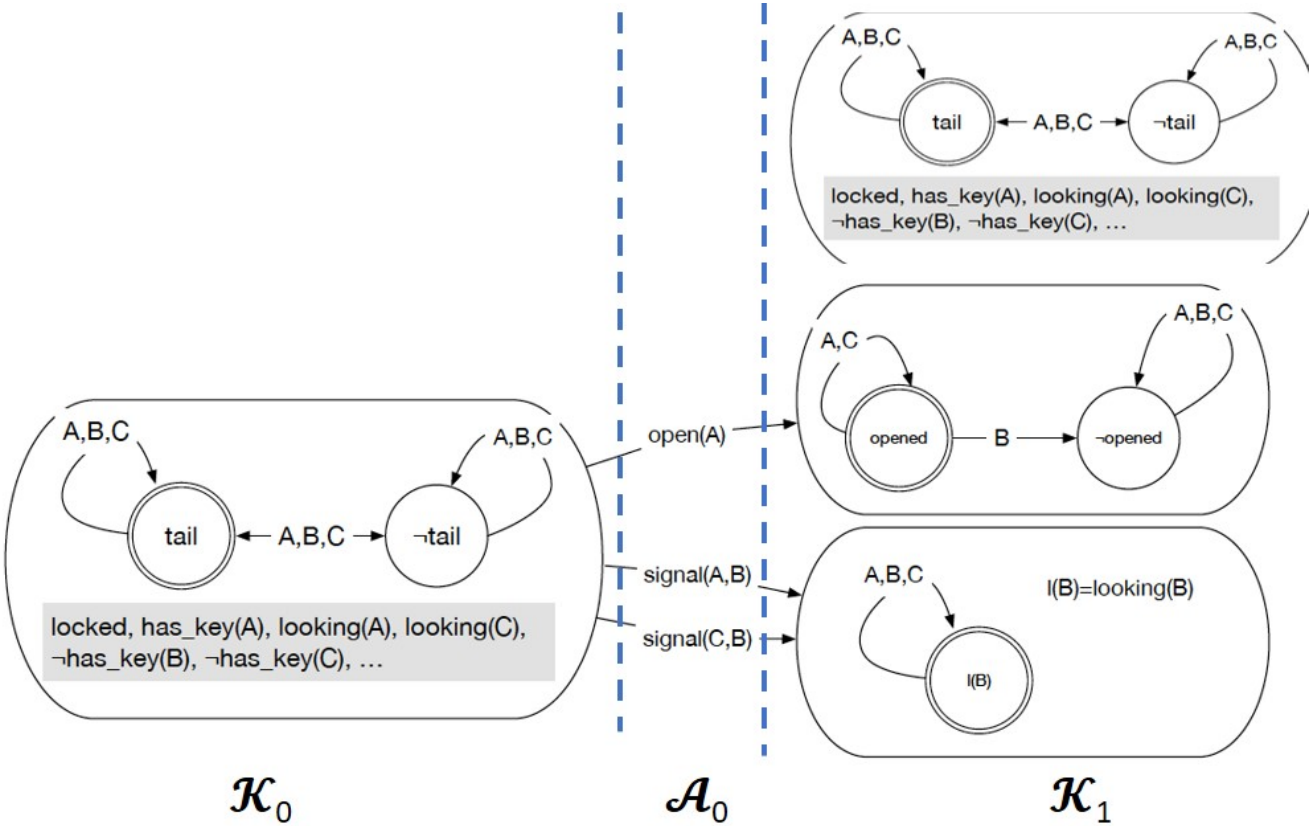
Forward Search Planner

1: **Input**: A planning problem $P = \langle F, \mathcal{AG}, A, O, s_0, \phi_g \rangle$
2: **Output**: A solution for P if exists; failed otherwise
3: Compute the initial state given s_0: (M_i, W_i)
4: Initialize a priority queue $q = [(\{(M_i, W_i)\}, [])]$
5: **while** q is not empty **do**
6: $(\Omega, plan) = dequeue(q)$
7: If $(M, W_d) \models \phi_g$ for every $(M, W_d) \in \Omega$ then return $plan$
8: **for** action a executable in every (M, W_d) in Ω **do**
9: Compute $\Omega' = \bigcup_{(M,W_d)\in\Omega} \Phi_D(a, (M, W_d))$
10: Compute heuristics and insert $(\Omega', plan \circ a)$ into q
11: **end for**
12: **end while**
13: **return failed**

Forward Search Planner

Epistemic Planning Graph

Alternation of Epistemic States Levels (K_i) and Action levels (A_i)[1]



$\mathcal{K}_i$ is a set of *incomplete* e-model where $\mathcal{K}_0$ is s

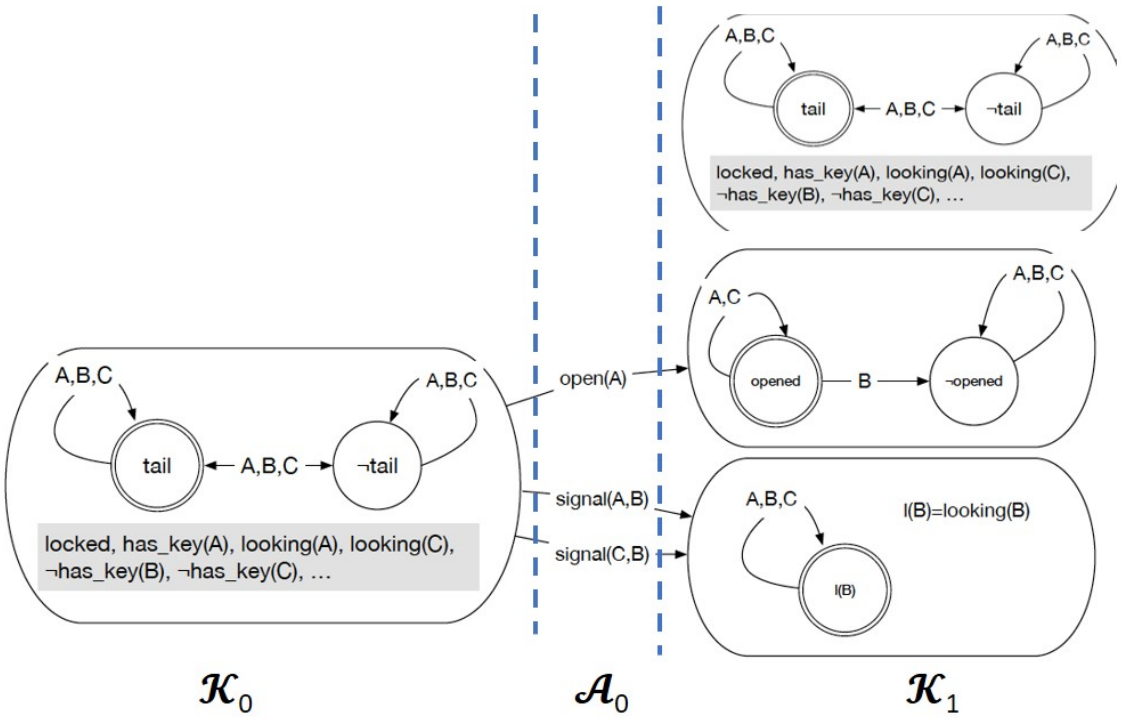
$\mathcal{A}_i$ is a set of actions *potentially applicable* in $\mathcal{K}_i$

[1] For simplicity focus on deterministic observable theories.

Forward Search Planner

Epistemic Planning Graph

K_i Possibly Entails φ



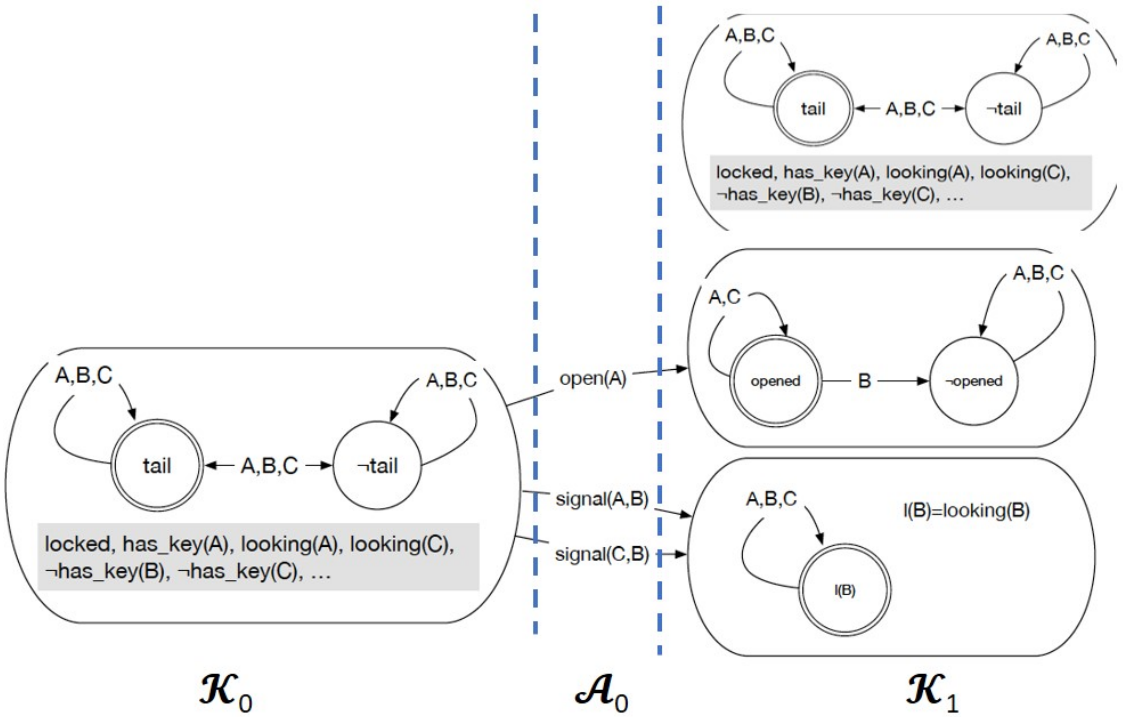
$\mathcal{K}_0 \models$ has_key(A) $\wedge$ $\boldsymbol{K}_A$ *has_key(A)*

$\mathcal{K}_1 \models$ opened $\wedge$ tail $\wedge$ looking(B)
$\mathcal{K}_1 \models$ opened $\wedge$ tail $\wedge$ looking(B)
$\mathcal{K}_1 \models \boldsymbol{K}_C$(opened $\wedge$ looking(B))

Forward Search Planner

Epistemic Planning Graph

A_i: Actions Potentially Applicable in K_i



executable *open(A)* **if** *has_key(A)* $\wedge$ $\boldsymbol{K}_A$ *has_key(A)*

$\mathcal{K}_0 \models$ has_key(A) $\wedge$ $\boldsymbol{K}_A$ *has_key(A)*
$\rightarrow$ *open(a)* $\in \mathcal{A}_0$

$\mathcal{K}_1 \models$ opened $\wedge$ tail $\wedge$ looking(B)
$\mathcal{K}_1 \models$ opened $\wedge$ tail $\wedge$ looking(B)
$\mathcal{K}_1 \models \boldsymbol{K}_C$(opened $\wedge$ looking(B))

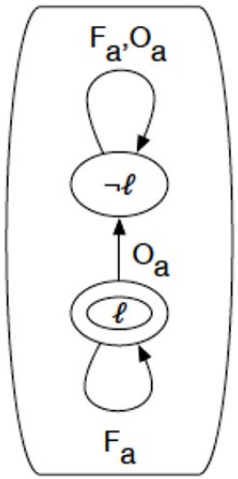
Forward Search Planner

Epistemic Planning Graph

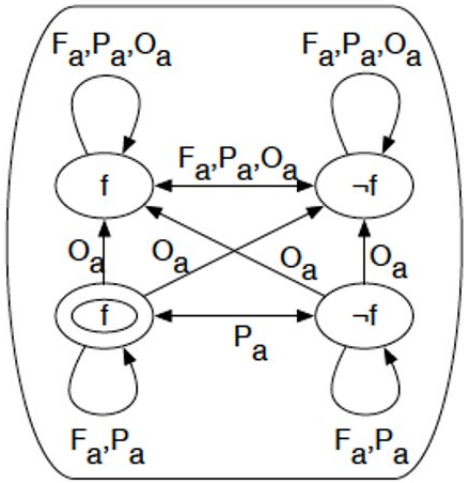
$$K_i = K_{i-1} \cup \bigcup_{a \in A_{i-1}} \textbf{Results}(a, K_i)$$

a **causes** ℓ; *i.e., a* is an ontic action
α **observes** *a; i.e.,* $\alpha \in F_a$; $O_a = \mathcal{AG} \setminus F_a$

a **determines** *f* ; i.e., *a* is a sensing action
a ***announces*** ℓ ; i.e., *a* is an announcement action
α **observes** *a* ; *i.e.,* $\alpha \in F_a$
β **aware_of** *a* ; *i.e.,* $\beta \in P_a$
; *i.e.,* $O_a = \mathcal{AG} \setminus (F_a \cup P_a)$



result of ontic action *a*

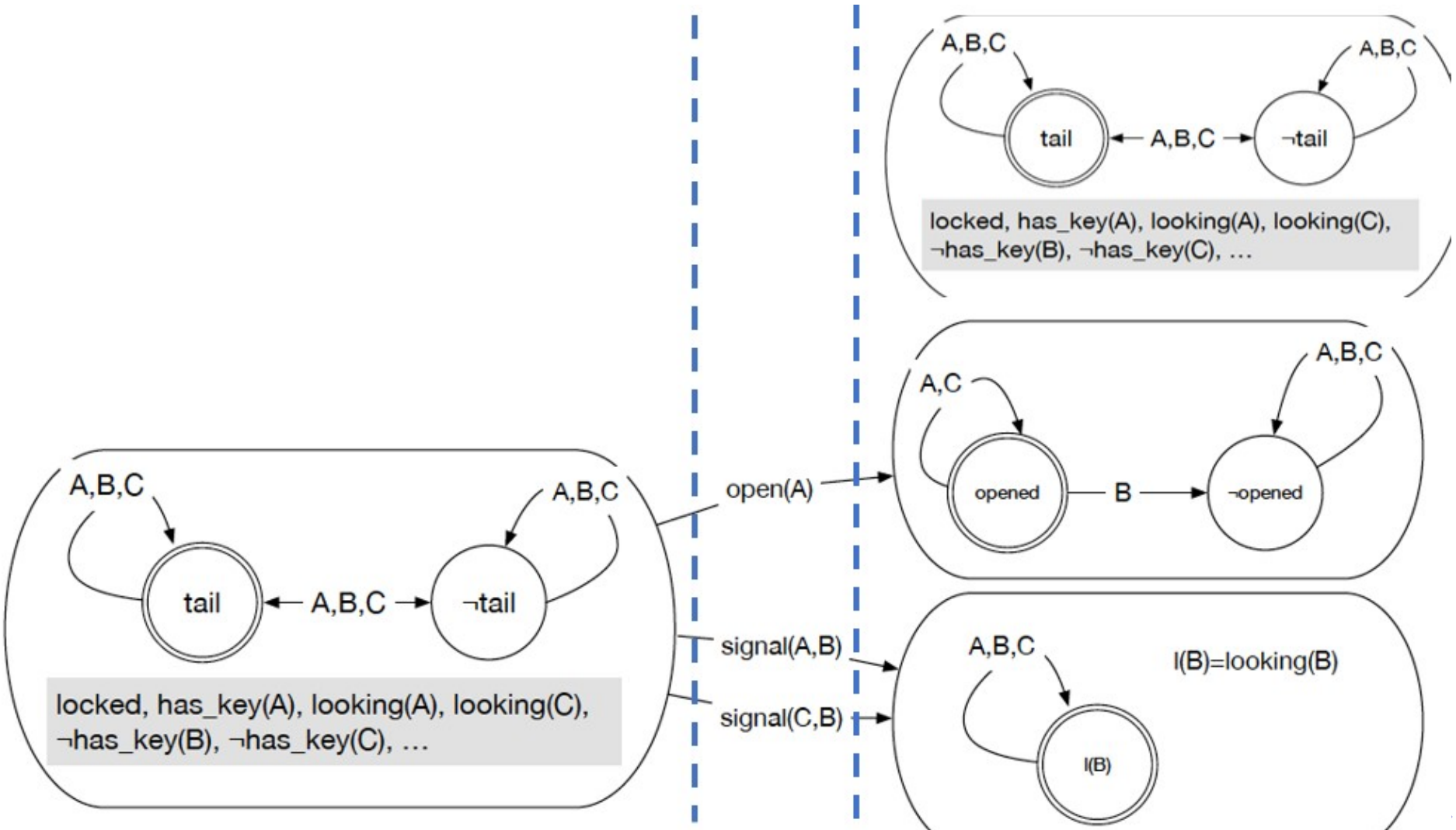


result of sensing (or announcement) action *a*

Forward Search Planner

Epistemic Planning Graph

Example



Forward Search Planner

Heuristics

$level(\phi)$ smallest level such that K_i possibly entails ϕ

Given $\phi_g = \phi_1 \wedge \cdots \wedge \phi_k$:

- $h^{max}(\phi_g) = max\{level(\phi_i) \mid 1 \leq i \leq k\}$
- $h^{sum}(\phi_g) = \sum_{i=1}^{k} level(\phi_i)$

Forward Search Planner

Some Experiments

Selective Communication: SC(3,4) $\lvert\mathcal{AG}\rvert = 3$, $\lvert F\rvert = 5$, $\lvert A\rvert = 7$						Selective Communication: SC(5,6) $\lvert\mathcal{AG}\rvert = 5$, $\lvert F\rvert = 7$, $\lvert A\rvert = 9$						Selective Communication: SC(7,8) $\lvert\mathcal{AG}\rvert = 7$, $\lvert F\rvert = 9$, $\lvert A\rvert = 11$					
L	d	MEPK	RP-MEP	EPF	PG-EFP	L	d	MEPK	RP-MEP	EFP	PG-EFP	L	d	MEPK	RP-MEP	EFP	PG-EFP
	1	.01	.1	.01	.02		1	.6	.2	.02	.04		1	35	.36	.22	TO
2	3	.2	.5	.02	.07	2	3	TO	3.2	.02	.04	5	3	TO	10.7	.22	TO
	5	TO	28	.03	.08		4	TO	51.58	.03	.04		4	TO	292	.24	TO
	1	.02	.1	.02	.06		1	.68	.2	.07	TO		1	35	.36	1.9	TO
3	3	.2	.5	.02	.07	4	3	TO	3.18	.08	TO	7	3	TO	10.8	1.92	TO
	5	TO	30	.02	.06		4	TO	54.78	.08	TO		4	TO	300	1.9	TO
	1	.05	.1	.08	TO		1	.81	.2	.51	.35		1	35.7	.32	23.7	1.86
5	3	.21	.6	.09	TO	6	3	TO	3.21	.52	.36	9	3	TO	12.72	24	1.9
	5	TO	28	.1	TO		4	TO	51.81	.51	.34		4	TO	312	23.5	1.93

Forward Search Planner

Some Experiments

Problem	L	EFP	PG-EFP
CC(2,2,3) $\lvert\mathcal{AG}\rvert = 2,\ \lvert F\rvert = 10,\ \lvert A\rvert = 16$	2	.61	.81
	5	48.6	2.5
	6	278.6	4.3
CC(2,2,4) $\lvert\mathcal{AG}\rvert = 2,\ \lvert F\rvert = 14,\ \lvert A\rvert = 22$	2	22.2	27.5
	4	TO	70
	7	TO	160
CC(3,2,3) $\lvert\mathcal{AG}\rvert = 3,\ \lvert F\rvert = 13,\ \lvert A\rvert = 24$	2	3.3	2.3
	5	257.9	7.9
	6	TO	10.3
CC(3,3,3) $\lvert\mathcal{AG}\rvert = 3,\ \lvert F\rvert = 12,\ \lvert A\rvert = 21$	2	.42	.68
	5	115.7	2.27
	6	TO	3

ASP Implementation [Pontelli et al. (2012)]

Representing Formulae and States

- τ translate formulae to terms, e.g.,

$$\tau(B_i(f \wedge \neg g)) = b(i, and(f, neg(g)))$$

- Representing state (M, s) at time T as facts
  - for each $u \in M[S]$:

$$st(u, T)$$

  - to identify start state:

$$real(s, T)$$

  - for each $(u, v) \in M[i]$

$$r(i, u, v, T)$$

  - if fluent literal ℓ $(M, u) \models \ell$:

$$h(\tau(\ell), u, T)$$

ASP Implementation

Entailment

Extend h

$$
\begin{array}{lcl}
h(or(A1, A2), S, T) & :- & h(A1, S, T). \\
h(or(A1, A2), S, T) & :- & h(A2, S, T).
\end{array}
$$

$$
\begin{array}{lcl}
n\_h(b(I, A), S, T) & :- & r(I, S, S1, T), \mathit{not}\ h(A, S1, T). \\
h(b(I, A), S, T) & :- & \mathit{not}\ n\_h(b(I, A), S, T).
\end{array}
$$

ASP Implementation

Initial State

m states in initial Kripke structure

$$1\{size(I) : between(0, I, m)\}1.$$
$$st(I, 0) :- I \geq 0, I \leq m.$$
$$0\{h(F, S, 0)\}1 :- st(S, 0), fluent(F).$$
$$0\{r(Ag, S1, S2, 0)\}1.$$
$$1\{real(S, 0) : st(S, 0)\}1.$$

ASP Implementation

Initial State

- For each **initially** φ

$$:- \mathit{real}(S,0), \mathit{not}\ h(\tau(\varphi),S,0).$$

- For each **initially** $C(\varphi)$

$$:- \mathit{real}(S,0), \mathit{reach}(S,S1,0), \mathit{not}\ h(\tau(\varphi),S1,0).$$

- For each **initially** $C(B_i\varphi \vee B_i\neg\varphi)$

$$:- \mathit{real}(S,0), \mathit{reach}(S,S1,0), r(i,S1,S2,0),$$
$$\mathit{not}\ h(\tau(\varphi),S2,0), \mathit{not}\ h(\tau(\neg\varphi),S2,0).$$

- For each **initially** $C(\neg B_i\varphi \wedge \neg B_i\neg\varphi)$

$$\mathit{not\_agree} :- \mathit{real}(S,0), \mathit{reach}(S,S1,0),$$
$$h(or(b(i,\tau(\varphi)),\tau(\neg\varphi)),S1,0).$$
$$:- \mathit{not\_agree}.$$

ASP Implementation

Initial State

Finally, ensure that the structure is S5:

$$
\begin{array}{lcl}
r(A,S,S,0) & :- & st(S,0), agent(A). \\
r(A,S1,S2,0) & :- & r(A,S2,S1,0). \\
r(A,S1,S3,0) & :- & r(A,S1,S2,0), r(A,S2,S3,0).
\end{array}
$$

ASP Implementation

Observability

- y **observes a if** φ

$$obs(y, \mathbf{a}, T) : -occ(\mathbf{a}, T), real(S, T), h(\tau(\varphi), S, T).$$

- y **partially_observes a if** φ

$$pobs(y, \mathbf{a}, T) : -occ(\mathbf{a}, T), real(S, T), h(\tau(\varphi), S, T).$$

- otherwise for each agent y and action instance $\mathbf{a}$

$$oth(y, \mathbf{a}, T) \quad : - \quad action(\mathbf{a}), occ(\mathbf{a}, T), not\ obs(y, \mathbf{a}, T), \\ not\ pobs(y, \mathbf{a}, T).$$

ASP Implementation

Transition Function

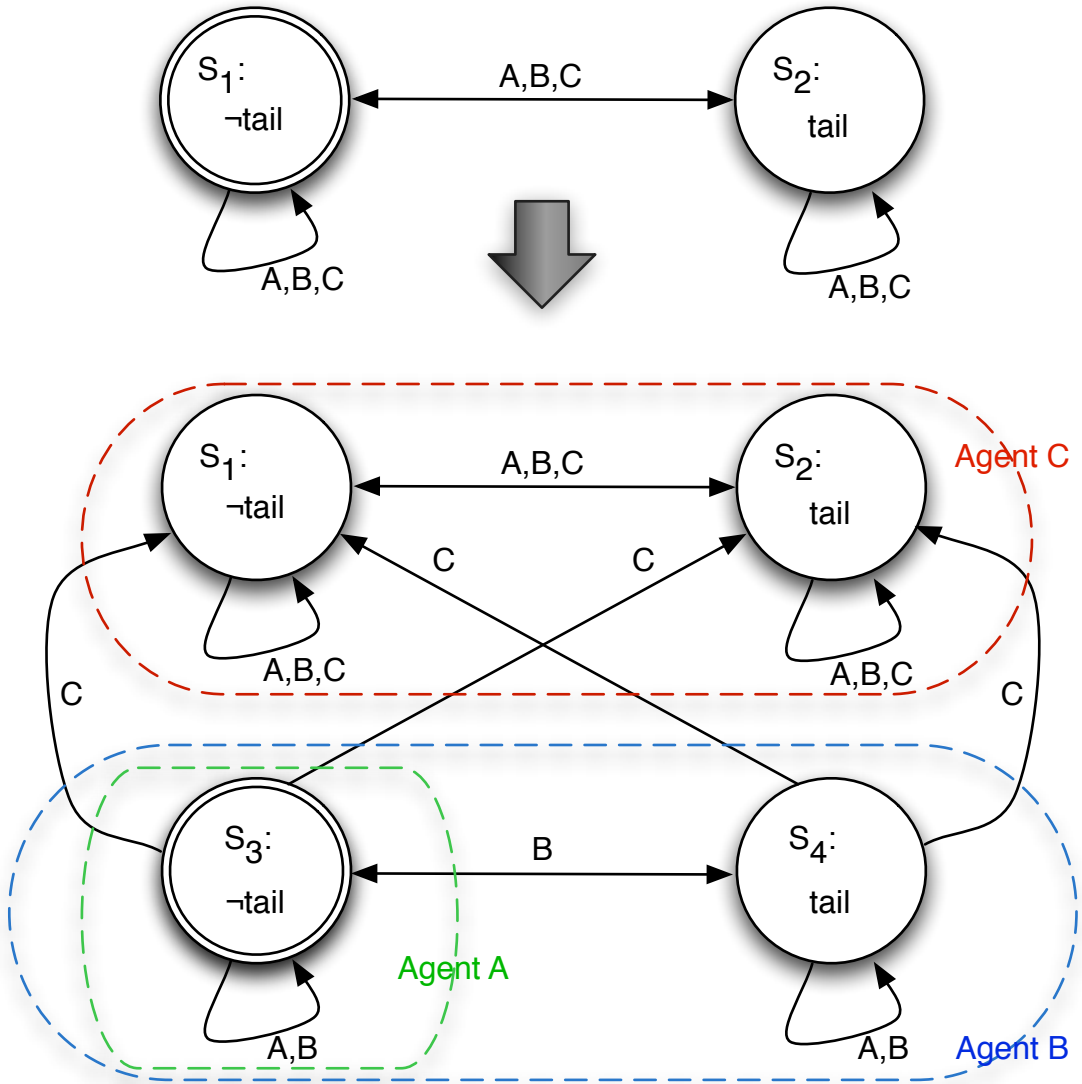
- **executable a if** ϖ

$$\begin{array}{lll} possible(\mathbf{a}, T) & :- & real(S, T), h(\tau(\varphi), S, T). \\ & :- & occ(\mathbf{a}, T), not\ possible(\mathbf{a}, T). \end{array}$$

ASP Implementation

Transition Function

Mechanical construction of new Kripke state



ASP Implementation

Transition Function – Sensing

Interpretation unchanged

$$
\begin{array}{lcl}
h(L, S, T+1) & :- & occ(\mathbf{a}, T), h(L, S, T). \\
h(L, S+M, T+1) & :- & occ(\mathbf{a}, T), h(L, S, T).
\end{array}
$$

Duplicate states

$$
\begin{array}{lcl}
st(S, T+1) & :- & occ(\mathbf{a}, T), st(S, T). \\
st(S+M, T+1) & :- & occ(\mathbf{a}, T), st(S, T). \\
real(S+M, T+1) & :- & occ(\mathbf{a}, T), real(S, T).
\end{array}
$$

ASP Implementation

Transition Function – Sensing

Maintain existing relations in original copy

$$r(A, S1, S2, T+1) :- occ(\mathbf{a}, T), r(A, S1, S2, T).$$

Fully observant agents have sensed fluent f

$$\begin{aligned} r(A, S1+M, S2+M, T+1) :- & obs(A, \mathbf{a}, T), occ(\mathbf{a}, T), \\ & r(A, S1, S2, T), h(f, S1, T), h(f, S2, T). \\ r(A, S1+M, S2+M, T+1) :- & obs(A, \mathbf{a}, T), occ(\mathbf{a}, T), \\ & r(A, S1, S2, T), h(neg(f), S1, T), h(neg(f), S2, T). \end{aligned}$$

Partially aware agents

$$\begin{aligned} r(A, S1+M, S2+M, T+1) :- & pobs(A, \mathbf{a}, T), occ(\mathbf{a}, T), \\ & r(A, S1, S2, T). \end{aligned}$$

Oblivious agents

$$\begin{aligned} r(A, S1+M, S2, T+1) :- & oth(A, \mathbf{a}, T), occ(\mathbf{a}, T), \\ & r(A, S1, S2, T). \end{aligned}$$

ASP Implementation

Planning

Generation of action sequences

$$1\,\{occ(A,T) : action\_instance(A)\}\,1 :- T < n.$$

Goal Satisfaction φ

$$:- real(S,n), h(\tau(\varphi),S,n).$$

1. Overview: a brief history of action languages **(20 mnt - Chitta)**
2. Action languages in single agent environments **(70 mnt - Son)**
   - The action language $\mathcal{A}$, state, and transition function
   - $\mathcal{AL}$: $\mathcal{A}+$ static causal laws, non-deterministic and sensing actions
   - GOLOG
   - Action languages: related approaches and planning
3. Action languages and causality **(30 mnt - Chitta)**
   - Pearl's do-calculus
4. Action languages in multi-agent environments **(60 mnt – Son)**
   - mA*, Kripke structure, update models, and transition function
   - mA* in epistemic planning
5. Action languages in commonsense reasoning **(18 mnt - Chitta)**
   - Commonsense reasoning and actions
   - Acquiring knowledge about actions
6. Open challenges, conclusion, and discussion **(12 mnt - Chitta)**

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 1: A Property May Prevent an Action

Sentence: The man couldn't lift his son because he (pronoun) was so weak.

Question: Who was weak?

Answer Choices: a) man b) son

Required Knowledge:
person1 is weak may prevent person1 lifts someone

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 2: An Action May Cause an Action

Sentence: The city councilmen refused the demonstrators a permit because they (pronoun) feared violence.

Question: Who feared violence?

Answers Choices: a) councilmen b) demonstrators

Required Knowledge:
group1 fears violence may cause group1 refuses permit

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 3: A Property May Cause an Action

Sentence: The sculpture rolled off the shelf because it (pronoun) was not anchored.

Question: What was not anchored?

Answer Choices: a) sculpture b) shelf

Knowledge Needed:
object1 is not anchored may cause object1 is rolled off

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 4: An Action May Cause a Property

Sentence: I took the water bottle out of the backpack so that it (pronoun) would be handy.

Question: What would be handy?

Answer Choices: a) bottle b) backpack

Required Knowledge:
object1 is taken out of something may cause object1 is handy

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 5: An Action May Prevent an Action

Sentence: Beth didn't get angry with Sally, who had cut her off, because she (pronoun) stopped and counted to ten.

Question: Who counted to ten?

Answers: a) Beth b) Sally

Required Knowledge:
person1 counts to ten may prevent person1 gets angry

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 6: An Action May be Followed By an Action

Sentence: The customer walked into the bank and stabbed one of the tellers. He was immediately taken to the hospital.

Question: Who was taken to the hospital?

Answers: a) teller b) customer

Required Knowledge:
person1 is stabbed may be followed by person1 is taken to hospital

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 7: An Action May be Followed by a Property

Sentence: Sam broke both his ankles and he is walking with crutches. But a month or so from now they (pronoun) should be unnecessary.

Question: What should be unnecessary?

Answer Choices: a) ankles b) crutches

Required Knowledge:
person1's ankles are broken and person1 walks with crutches may be followed by crutches are unnecessary

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 8: A Property May be followed by an Action

Sentence: Thomson visited Cooper's grave in 1765. At that date he (pronoun) had been dead for five years.

Question: Who had been dead for five years?

Answer Choices: a) Cooper b) Thomson

Knowledge Needed:
person1 is dead may be followed by person1's grave is visited

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 9: A Property May Cause a Property

Sentence: Sam and Amy are passionately in love, but Amy's parents are unhappy about it, because they (pronoun) are fifteen.

Question: Who are fifteen?

Answer Choices: a) Sam and Amy b) Amy's parents

Knowledge Needed: person1 is in love and person1 is fifteen years old may cause person1's parents are unhappy

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 10: Action-action suggestions

Sentence: Steve follows Fred's example in everything. He influences him (pronoun) hugely.

Question: Who is influenced?

Answer Choices: a) Steve b) Fred

Knowledge Needed: person1 follows person2's example in everything may suggest person1 is influenced by person2

Some commonsense knowledge types centered around actions

Action knowledge types from Winograd examples

Action examples and commonsense inferences that can be made from them

Knowledge Type 11: Action-property suggestions

Sentence: The fish ate the worm. It (pronoun) was hungry.

Question: What was hungry?

Answer Choices: a) fish b) worm

Knowledge Needed: animal1 eats something may co-occur with animal1 is hungry

Some Commonsense knowledge types centered around actions

Agent X does action A on/with-respect-to(w.r.t.) agent Y				
Action attribute	Condition	w.r.t. X	w.r.t. Y	w.r.t. Others
Conditional effect of this action	condition	effect of A on X's attributes (xEffect, xReact)	effect of A on Y's attributes (oEffect, oReact)	effect of A on the environment (oEffect, oReact)
	condition	triggers another action by X (xWant)	triggers another action by Y (oWant)	triggers another action by another agent (oWant)
Executability Condition		On X	On Y	On other agents or objects in the environment
Reflective Condition		On X (xAttr)	On Y	On other agents or objects in the environment
Triggering or Preceding actions		Another action by X (xNeed)	An action by Y	An action by another agent
		About X's properties	About Y's properties	About properties of other agents
Preventing conditions		About X's attributes	About Y's attributes	About other agents or objects
Preventing actions		Another action by X	An action by Y	An action by another agent
Motivation behind the action		For a desired property of X (xIntent)	For a desired property of Y	For a desired property of the world
		To trigger another action by X	To trigger another action by Y	To trigger action by another agent

Action examples and commonsense inferences that can be made from them

Event	Inference Examples	Inference aspect
Person X pays Person Y a complement	Person X wanted to be nice	xIntent
	Person X will feel good	xReact
	Person Y will feel flattered	oReact
	Person X will want to chat with Person Y	xWant
	Person Y will smile	oEffect
	Person Y will complement Person X back	oWant
	Person is flattering	xAttr
Person X makes Person Y's coffee	Person X needs to put the coffee in the filter	xNeed
	Person X gets thanked	xEffect
Person X shakes hand with Person Y	Person Y's hand is extended	New
	Person Y extends the hand	New
Person X throws a ball to Y	Person X had a ball	New
Person X marries Person Y	X and Y are married	New
Person X asks help from Person Y	Person Y is approachable	New

Additional kinds of commonsense knowledge centered around actions

- Initiation and termination conditions of non instantaneous (and possibly continuous) actions such as "falling" and "driving".
- Action modifiers: Commonsense knowledge describing the effect of "eating", "eating a little" and "eating a lot" can be different.
- Actions may have additional attributes such as implements used in the action or the objects that are target of the action, and commonsense knowledge about such actions may depend on the specific implement used for that action or the specific target of the action.
  - The effect of hitting a ball is different from hitting a wall which is different from hitting a car.
  - Hitting with a bat is different from hitting with a sledgehammer which is different from hitting with a feather.

Additional kinds of commonsense knowledge centered around actions

- Actions together with expectations about those actions
  - "Going to sleep" is expected to follow someone feeling sleepy.
  - X feels sleepy but continues to work implies X may have a deadline.
  - An action A by X may have many possible reaction by Y, and the particular reaction that takes place may shed additional light such as on X's attributes or Y's attributes: If X attacks Y and Y cries then the commonsense conclusion about Y would be different from if X attacks Y and Y counterattacks or if Y ducks.

Additional kinds of commonsense knowledge centered around actions

- Commonsense knowledge about "perceptions" or sensing actions
  - IF X observes/perceives Z then it will do A,
  - IF X observes/perceives Z then it will/may conclude event E will happen'
  - IF X observes/perceives Z then it will conclude C
  - If X sees lightning then he will conclude that he will hear thunder soon
  - If X sees a carjacking he will call 911
  - If X hears his baby crying he will console the baby
  - If X sees a riot happening he will run away and call police
- Commonsense knowledge about "perceptions" followed by an action
  - If X touches his kid's head and it is burning hot he will go to ER
  - If X opens the door and sees a package lying then he will bring it inside

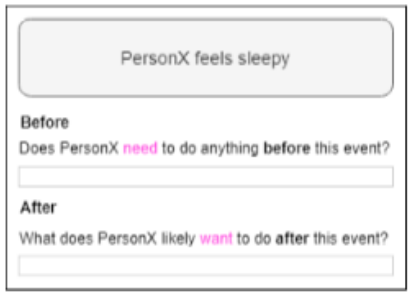
Additional kinds of commonsense knowledge centered around actions

- Commonsense knowledge that associate modalities (intentions, beliefs, and obligations, etc.) with an agent's action performed unilaterally or in response to some perceptions
  - X put water on a fire implies X believed that the water will put out the fire
  - X put water on a fire implies X intended to put out the fire
  - X does not like to do A, but does it every week without being forced implies X feels obligated to do A

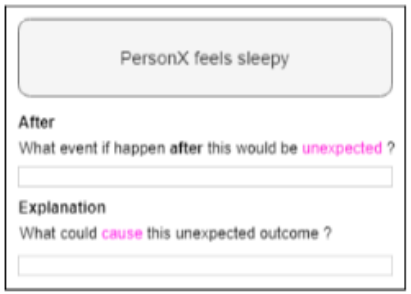
Knowledge acquisition through crowdsourcing: General technique

- Crowdsourcing commonsense knowledge involves designing a well defined question answering task.
- The question answering task contains a large number of input problems that are automatically created using existing machine readable resources (e.g. Google NGram Corpus, existing knowledge bases, VerbNet).
- Each input problem normally contains a scenario (normally one or two sentences or an image) and a set of easy-to-follow questions from the context, which are given to the crowd workers.
- The crowd workers answers those questions.
- Those answers, questions and the original context are then combined together to form a large commonsense knowledge base.

Knowledge acquisition through crowdsourcing in ATOMIC



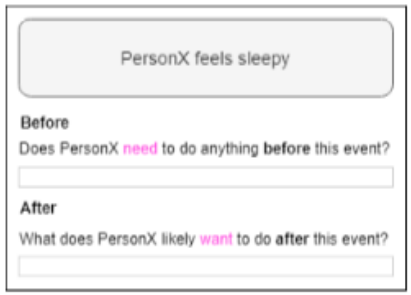
(a) A simplified example of a Crowdsourcing input prompt that was used to create the ATOMIC knowledge base.



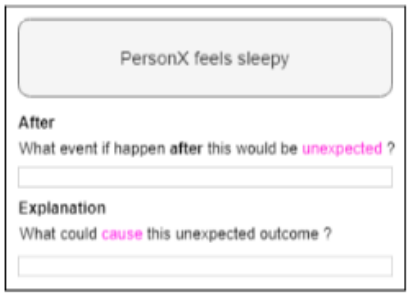
(b) An example of a input prompt that could be used to collect knowledge about unexpected scenarios.

- The descriptions of events ("PersonX feels sleepy") are automatically extracted machine readable resources such as stories, books, Google Ngrams, and Wiktionary idioms.
- A fixed set of manually crafted questions are asked to get information about these actions. For example, the question "What does PersonX most likely want to do after this event?" seeks knowledge about possible effects.
- But the current work does not ask some important questions.

Going beyond ATOMIC: An example to collect knowledge about unexpected scenarios.



(a) A simplified example of a Crowdsourcing input prompt that was used to create the ATOMIC knowledge base.



(b) An example of a input prompt that could be used to collect knowledge about unexpected scenarios.

- Input may describe "PersonX feels sleepy" with an optional list of events that normally follows the scenario (e.g. "PersonX goes to bed", "PersonX closes PersonX's eyes") and
- then ask the crowd worker to describe what might be an abnormal outcome (e.g. " PersonX continues working") and some most probable explanations (e.g. "PersonX has an impending deadline")

1. Overview: a brief history of action languages **(20 mnt - Chitta)**
2. Action languages in single agent environments **(70 mnt - Son)**
   - The action language $\mathcal{A}$, state, and transition function
   - $\mathcal{AL}$: $\mathcal{A}+$ static causal laws, non-deterministic and sensing actions
   - GOLOG
   - Action languages: related approaches and planning
3. Action languages and causality **(30 mnt - Chitta)**
   - Pearl's do-calculus
4. Action languages in multi-agent environments **(60 mnt – Son)**
   - mA*, Kripke structure, update models, and transition function
   - mA* in epistemic planning
5. Action languages in commonsense reasoning **(18 mnt - Chitta)**
   - Commonsense reasoning and actions
   - Acquiring knowledge about actions
6. Open challenges, conclusion, and discussion **(12 mnt - Chitta)**

Summary

- Action and change has been thought about from pre-Plato, Aristotle days
- AI approach to action and change has its foundations from Leibniz and Newton (frame problem)
- Since the early days of AI there has been a lot of progress in reasoning about action and change and planning - many theories and many systems
- But all assume input is given in a formal way
- Actions also play an important role in Probabilistic reasoning and statistical inference: But currently only a limited form of action do(l) - meaning making l true - is being considered

Challenges

- Challenge 1: How to consider more general actions in the context of statistical inference
- Natural language understanding (NLU) and Question answering (QA) has a symbiotic relationship with research in reasoning about actions (RAC)
  1. Research in RAC is useful in Natural language QA (NLQA)
  2. NLQA suggest many new challenges in RAC
- Challenge 2: Using research in RAC for better QA and NLU
- Challenge 3: Address RAC issues suggested by NLQA
- Challenge 4: Lot of issues in RAC and planning in the multi-agent domain

Some Interesting directions in multi-agent domains

1. Deception:
   - Announcing φ implies $(M, s) \models \varphi$
   - Lies: announce φ such that $(M, s) \models \neg\varphi$
   - Bullshit: announce φ such that $(M, s) \not\models \varphi$ and $(M, s) \not\models \neg\varphi$
2. Static Causal Laws
3. Alternative semantic characterizations
   - Non-well-founded set theory


Thank You !

References I

Babb, J. and Lee, J. (2015). Action language BC+: preliminary report. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1424–1430.

Baral, C. and Gelfond, M. (2000). Reasoning agents in dynamic domains. In Minker, J., editor, *Logic-Based Artificial Intelligence*, pages 257–279. Kluwer Academic Publishers.

Baral, C., Gelfond, M., and Provetti, A. (1997). Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243.

Baral, C., McIlraith, S., and Son, T. C. (2000). Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proceedings of the Seventh International Conference on Principles of Knowledge and Representation and Reasoning (KR'2000)*, pages 311–322.

References II

Baral, C., Son, T. C., and Tuan, L.-C. (2002a). A transition function based characterization of actions with delayed and continuous effects. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR'2002)*, pages 291–302. Morgan Kaufmann Publisher.

Baral, C., Tran, N., and Tuan, L.-C. (2002b). Reasoning about actions in a probabilistic setting. In *AAAI/IAAI*, pages 507–512.

De Giacomo, G., Lespérance, Y., and Levesque, H. (2000). *ConGolog*, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2):109–169.

Fikes, R. and Nilson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208.

Gelfond, M. and Lifschitz, V. (1993). Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323.

References III

Gelfond, M. and Lifschitz, V. (1998). Action Languages. *Electronic Transactions on Artificial Intelligence*, 3(6).

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL — the Planning Domain Definition Language. Version 1.2. Technical Report CVC TR98003/DCS TR1165, Yale Center for Comp, Vis and Ctrl.

Giunchiglia, E., Kartha, G., and Lifschitz, V. (1997). Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–443.

Giunchiglia, E. and Lifschitz, V. (95). Dependent fluents. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1964–1969. Morgan Kaufmann Publishers, San Mateo, CA.

Kartha, G. and Lifschitz, V. (1994). Actions with indirect effects (preliminary report). In *KR 94*, pages 341–350.

Kowalski, R. and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67–95.

References IV

Le, T., Fabiano, F., Son, T. C., and Pontelli, E. (2018). EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains. In *International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press.

Lee, J. and Lifschitz, V. (2003). Describing additive fluents in action language C+. In *Proceedings of IJCAI*, pages 1079–1084.

Lee, J., Lifschitz, V., and Yang, F. (2013). Action language BC: preliminary report. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 983–989.

Lee, J. and Wang, Y. (2018). Weight learning in a probabilistic extension of answer set programs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018.*, pages 22–31.

References V

Levesque, H., Reiter, R., Lesperance, Y., Lin, F., and Scherl, R. (1997). GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–84.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh.

Pontelli, E., Son, T. C., Baral, C., and Gelfond, G. (2012). Answer set programming and planning with knowledge and world-altering actions in multiple agent domains. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, pages 509–526.

Son, T. C. and Baral, C. (2001). Formalizing sensing actions - a transition function based approach. *Artificial Intelligence*, 125(1-2):19–91.

References VI

Son, T. C., Pontelli, E., Baral, C., and Gelfond, G. (2014). Finitary s5-theories. In Fermé, E. and Leite, J., editors, *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, pages 239–252. Springer.

Thiebaux, S., Hoffmann, J., and Nebel, B. (2003). In Defense of PDDL Axioms. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*.

Thielscher, M. (2000). The Fluent Calculus: A Specification Language for Robots with Sensors in Nondeterministic, Concurrent, and Ramifying Environments. Technical Report CL-2000-01, Computational Logic Group, Department of Computer Science, Dresden University of Technology.