CSC 481: First Order Logic

c- Pragmatics and conclusion

Rodrigo Canaan
Assistant Professor
Computer Science Department
Cal Poly, San Luis Obispo
rcanaan@calpoly.edu

The basics

A language is composed of:

- **Syntax:** what symbols may be used and what combinations of symbols are well-formed
  - “I drove to work today” is a well-formed English sentence
  - “I work to today drove” is not
- **Semantics:** what each sentence means
  - Sentences are used to convey that the world is one way and not another
- **Pragmatics**: what the language is used for
  - The sentence “fire!” suggests a different response in a crowded theater or in a shooting range

The basics

A language is composed of:

- **Syntax:** what symbols may be used and what combinations of symbols are well-formed
  - "I drove to work today" is a well-formed English sentence
  - "I work to today drove" is not
- **Semantics:** what each sentence means

Last class

  - Sentences are used to convey that the world is one way and not another
- **Pragmatics**: what the language is used for
  - The sentence "fire!" suggests a different response in a crowded theater or in a shooting range

Satisfaction and Models

Assume that $t_1, \ldots, t_n$ are terms, P is a predicate of arity n, α and β are formulas, and x is a variable.

1. $\mathfrak{I}, \mu \models P(t_1, \ldots, t_n)$ iff $\langle d_1, \ldots, d_n \rangle \in \mathcal{P}$, where $\mathcal{P} = \mathcal{I}[P]$, and $d_i = \|t_i\|_{\mathfrak{I},\mu}$;
2. $\mathfrak{I}, \mu \models t_1 = t_2$ iff $\|t_1\|_{\mathfrak{I},\mu}$ and $\|t_2\|_{\mathfrak{I},\mu}$ are the same element of $\mathcal{D}$;
3. $\mathfrak{I}, \mu \models \neg\alpha$ iff it is not the case that $\mathfrak{I}, \mu \models \alpha$;
4. $\mathfrak{I}, \mu \models (\alpha \wedge \beta)$ iff $\mathfrak{I}, \mu \models \alpha$ and $\mathfrak{I}, \mu \models \beta$;
5. $\mathfrak{I}, \mu \models (\alpha \vee \beta)$ iff $\mathfrak{I}, \mu \models \alpha$ or $\mathfrak{I}, \mu \models \beta$ (or both);
6. $\mathfrak{I}, \mu \models \exists x.\alpha$ iff $\mathfrak{I}, \mu' \models \alpha$, for some variable assignment μ' that differs from μ on at most x;
7. $\mathfrak{I}, \mu \models \forall x.\alpha$ iff $\mathfrak{I}, \mu' \models \alpha$, for every variable assignment μ' that differs from μ on at most x.

1. married(‘Ryan’,’Blake’)
2. spouse(‘Ryan’) = ‘Blake’
3. ¬ dog(‘Garfield’)
4. dog(‘Toto’) ^ married(‘Ryan’,’Blake’)
5. dog(‘Garfield’) v dog(‘Toto’)
6. $\exists x$ married(x, ‘Blake’)
7. $\forall x \; dog(x) \rightarrow mammal(x)$

We say $\mathfrak{I} \models \alpha$ or α is satisfied in $\mathfrak{I}$ or (conversely) $\mathfrak{I}$ satisfies α

The basics

A language is composed of:

- **Syntax:** what symbols may be used and what combinations of symbols are well-formed
  - “I drove to work today” is a well-formed English sentence
  - “I work to today drove” is not
- **Semantics:** what each sentence means
  - Sentences are used to convey that the world is one way and not another
- **Pragmatics**: what the language is used for
  - The sentence “fire!” suggests a different response in a crowded theater or in a shooting range

Today

Pragmatics

Our goal

What do we want a KB for?

- Given a set of logical sentences assumed to represent relevant knowledge of the world (a knowledge base, or KB), we may want to:
  - Use the KB for querying: does some other fact follow from our KB?
    - Example: given the facts below, does if follow that Bob is the killer?
      - $killer(Alice) \lor killer(Bob)$
      - $hasAlibi(Alice)$
      - $\forall x : hasAlibi(x) \to \neg killer(x)$

Our goal

What do we want a KB for?

- Given a set of logical sentences assumed to represent relevant knowledge of the world (a knowledge base, or KB), we may want to:
  - Use the KB for as basis for an agent: what is the best action to do?
    - Example: given a certain Minesweeper board, which hidden tile should we explore next?
    - This can be achieved in several ways:
      - we could query for $safe(tile)$ and have our agent click that tile if so
      - or we could define a predicate $bestTileToExplore/1$ and some rule to produce it.

Our goal

What do we want a KB for?

- In both cases, we want the retrieve answer/action to “logically follow” from the KB
  - But what does this mean?

Logical Consequence

Intuition

- Given an interpretation, we can determine the truth value of any sentence
- However, our program has no idea what our intended interpretation is
- We would still like our program to be able to derive conclusions that agree with our interpretation

Logical Consequence

Intuition (cont.)

- Given the sentences:
  - S1 = “Fido is a dog”
  - S2 = “All dogs are mammals”
  - α = “Fido is a mammal”
- Each of these sentences may or may not be true for a given interpretation $\mathfrak{I}$
- What we can do is say: if $\mathfrak{I}$ happens to satisfy both S1 and S2, then it <u>must</u> satisfy α

Logical Consequence
Valid and unsatisfiable sentences

- Note also that the meaning of some sentences does not depend on any interpretation. For example, if α is any sentence:
  - $\alpha \vee \neg\alpha$ is always true (α is valid)
  - $\alpha \wedge \neg\alpha$ is always false (α is unsatisfiable)
- Valid sentences are also called tautologies
- Invalid sentences are also called contradictions

Logical Consequence

Definition

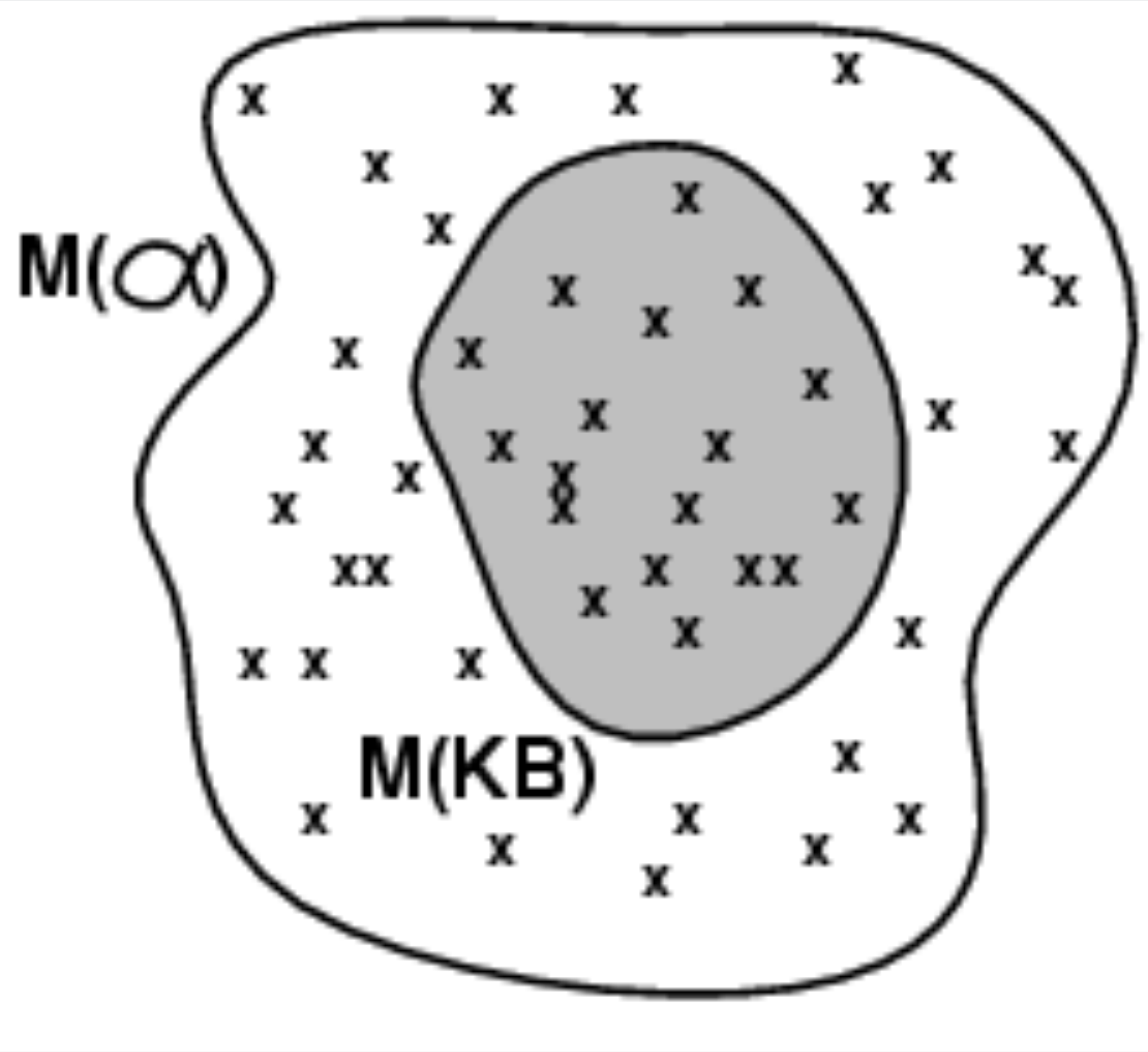
- Given a (finite) set of sentences $S = \{s_1, s_2, s_3, .., s_n\}$ and a sentence α, the following statements are equivalent:
  - S $\vDash \alpha$ (read "**S entails** α")
  - For every $\mathfrak{I}$, if $\mathfrak{I} \vDash$ S, then $\mathfrak{I} \vDash \alpha$
  - The set of interpretations that satisfy S is a subset of the set of interpretations that satisfy α
  - There is no $\mathfrak{I}$ such that $\mathfrak{I} \vDash$ S $\cup \{\neg\alpha\}$
  - S $\cup \{\neg\alpha\}$ is unsatisfiable
  - $(s_1 \wedge s_2 \wedge s_3 \wedge ... \wedge s_n) \rightarrow \alpha$ is a tautology
  - $(s_1 \wedge s_2 \wedge s_3 \wedge ... \wedge s_n) \wedge \neg\alpha$ is a condradiction

Models

- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated

- We say m is a model of a sentence α if α is true in m

- $M(\alpha)$ is the set of all models of α



- **Then KB $\models$ α iff $M(KB) \subseteq M(\alpha)$**

E.g.
- KB = Giants won and Reds won
- α = Giants won

CAL POLY

http://aima.eecs.berkeley.edu/slides-ppt/

Models

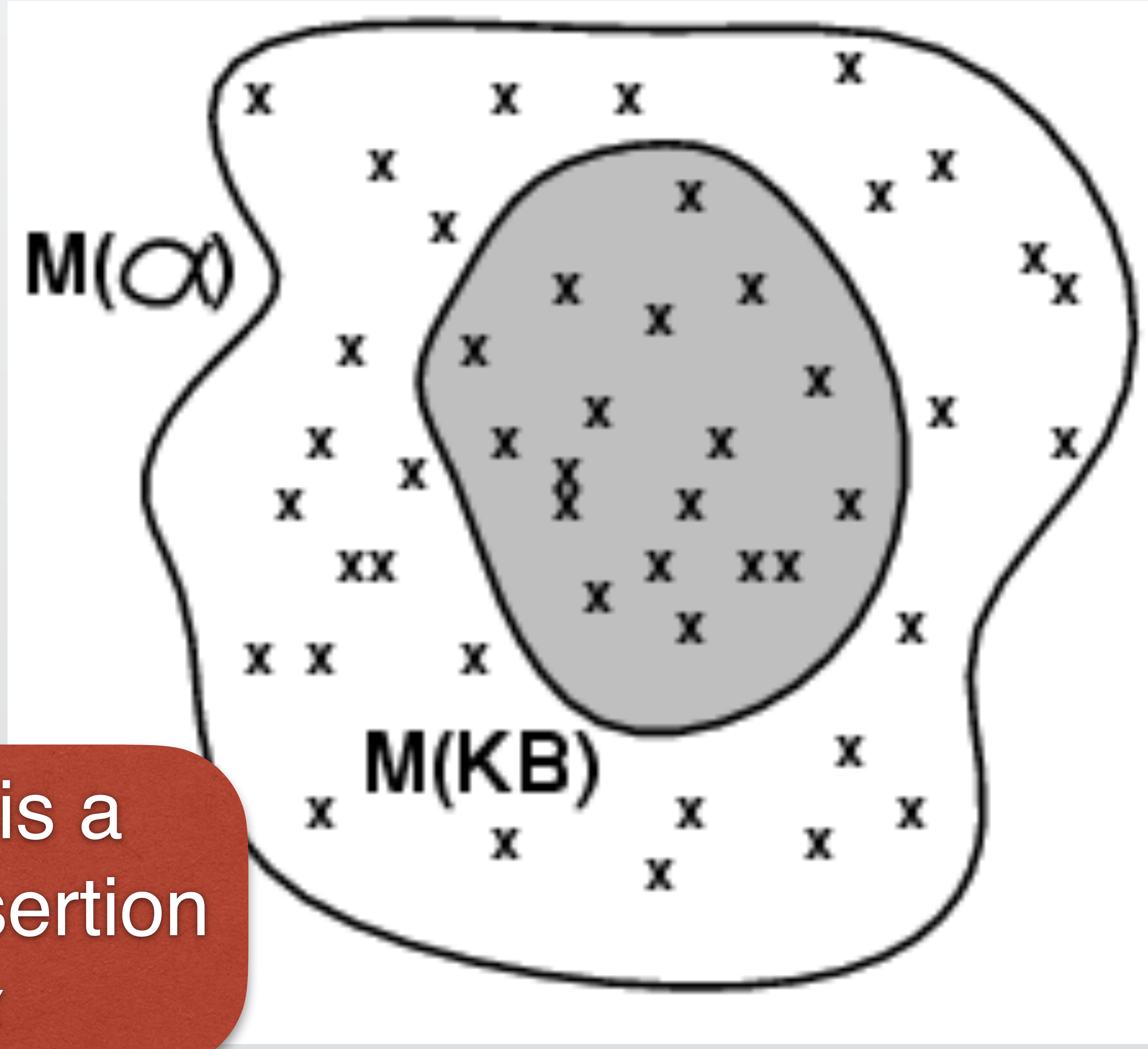
- **Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated**

- **We say *m* is a model of a sentence α if α is true in *m***

- *M(α)* is the set of all models of α

❖ **Then KB $\models$ α iff *M(KB)* $\subseteq$ *M*(α)**

E.g.
- *KB* = Giants won and Reds won
- α = Giants won



Note: KB is a stronger assertion than α

CAL POLY

http://aima.eecs.berkeley.edu/slides-ppt/

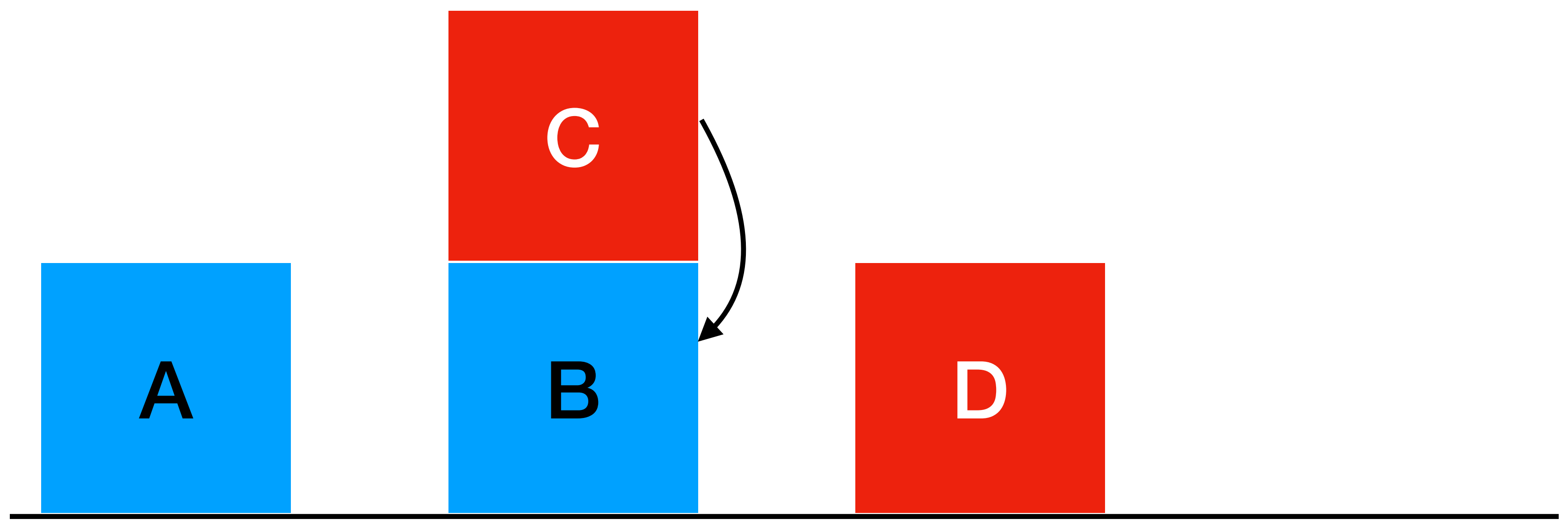
Example: block world

- An interpretation corresponds to an arrangement of blocks
  - Domain: a set of blocks
  - Predicates: blue/1, onTopOf/2

Example: block world

- An interpretation corresponds to an arrangement of blocks
  - Domain: a set of blocks
  - Predicates: blue, onTopOf

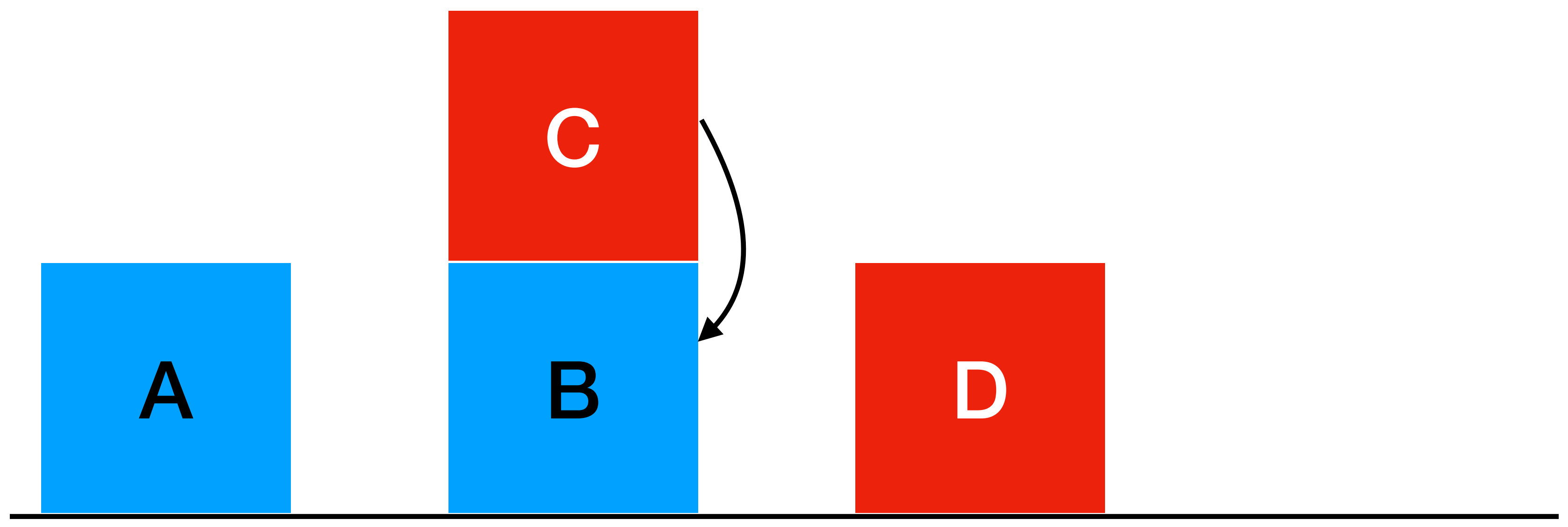
A possible interpretation



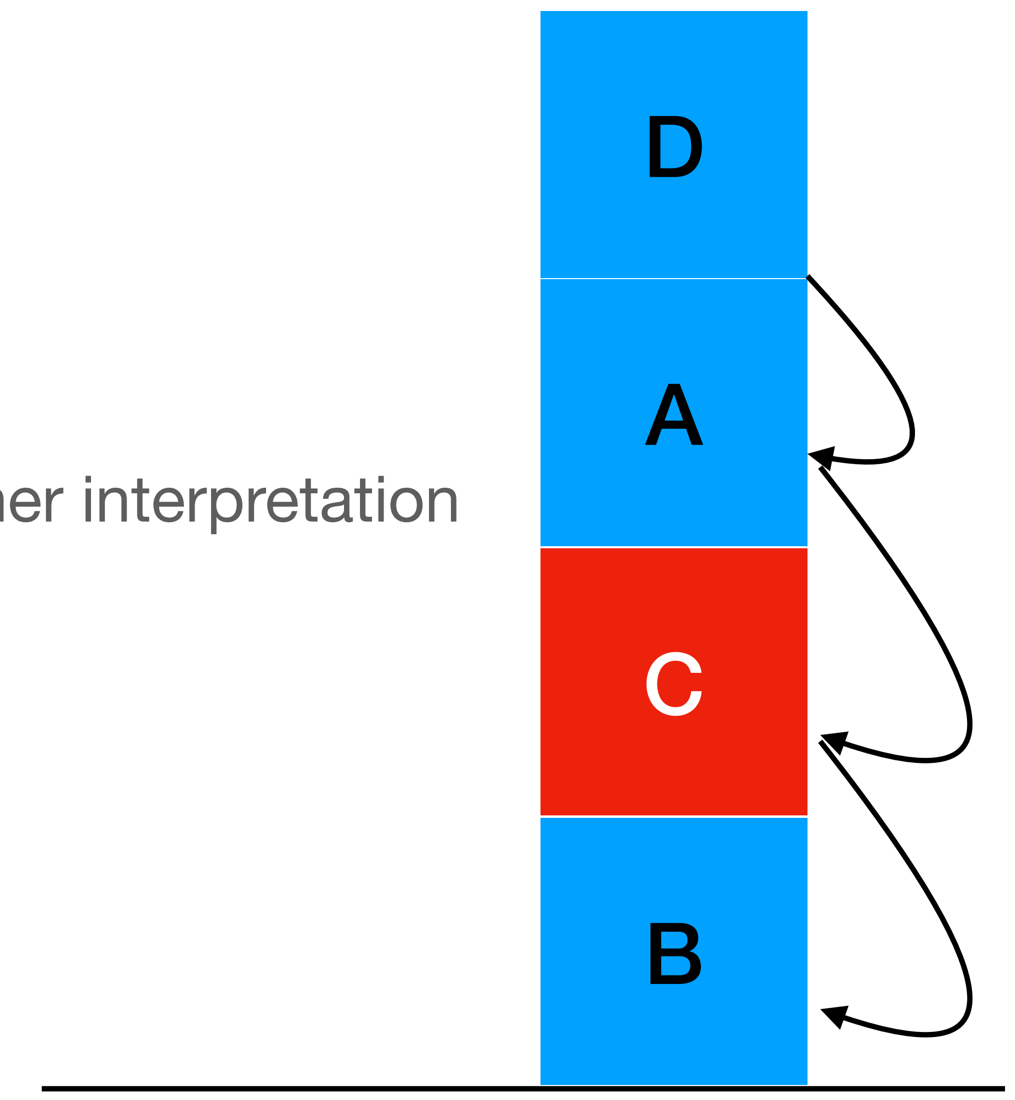
Example: block world

- An interpretation corresponds to an arrangement of blocks
  - Domain: a set of blocks
  - Predicates: blue, onTopOf

A possible interpretation



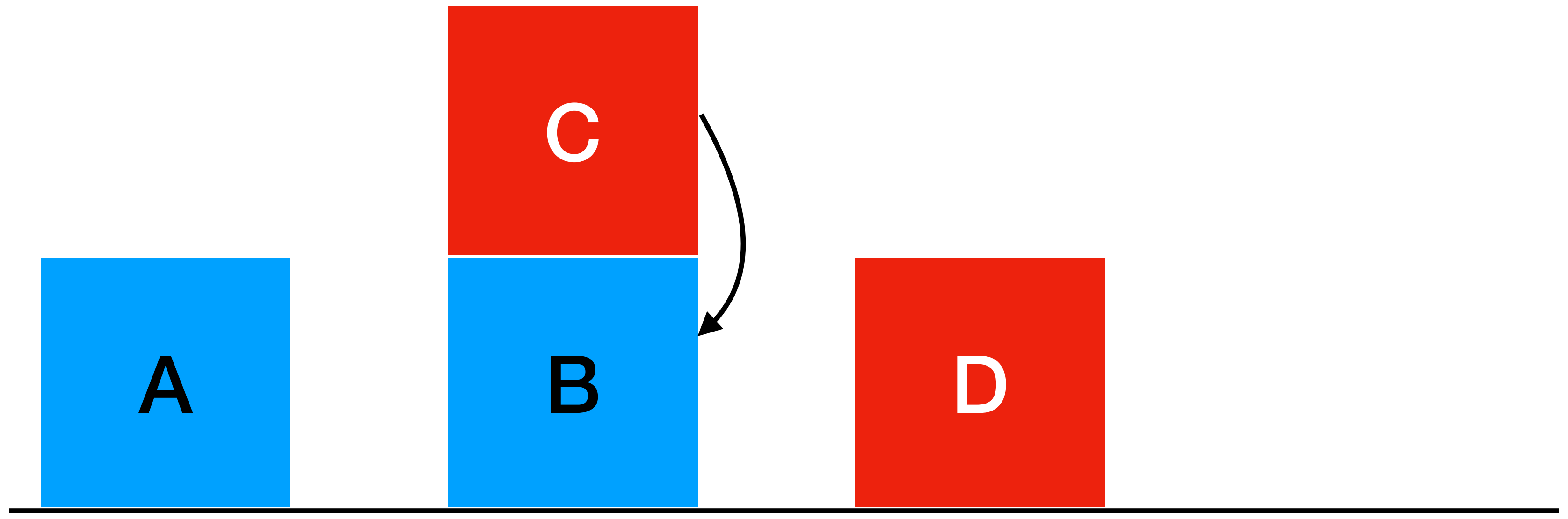
Another interpretation



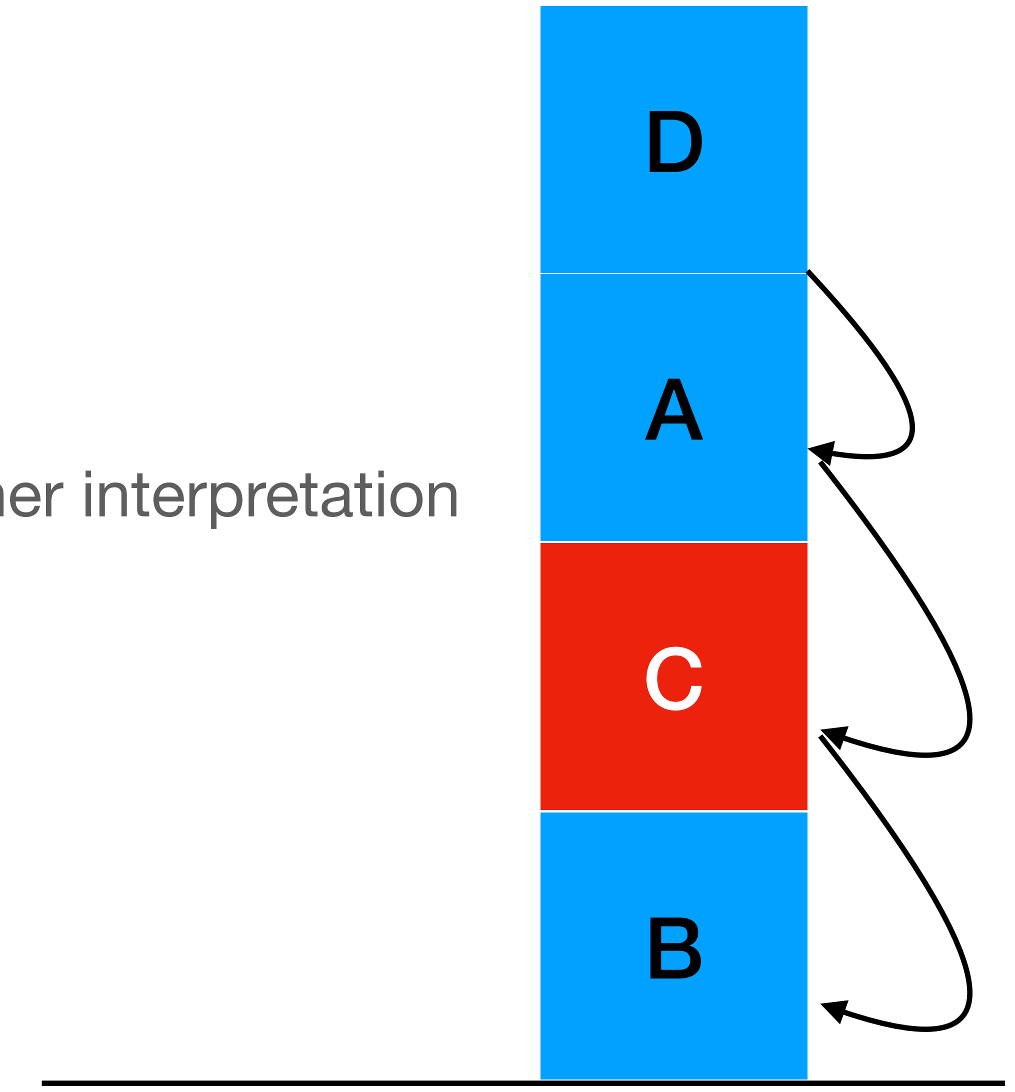
Example: block world

- An interpretation corresponds to an arrangement of blocks
  - Domain: a set of blocks
  - Predicates: blue, onTopOf

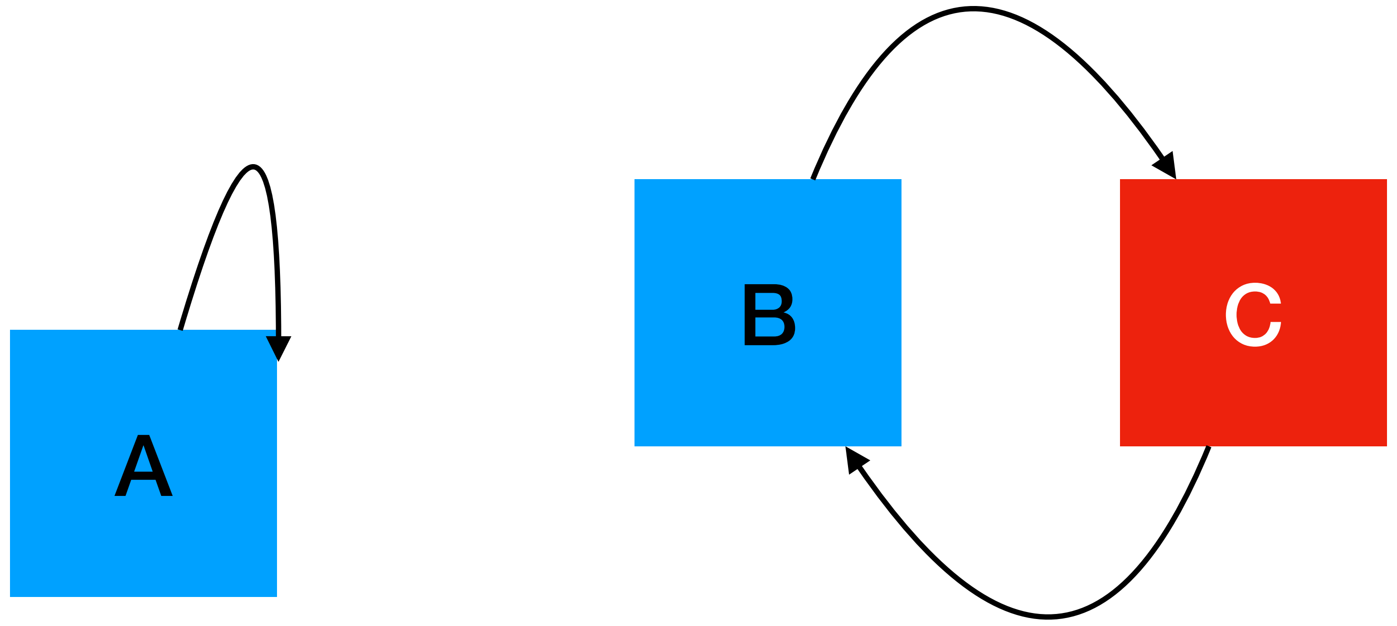
A possible interpretation



Another interpretation

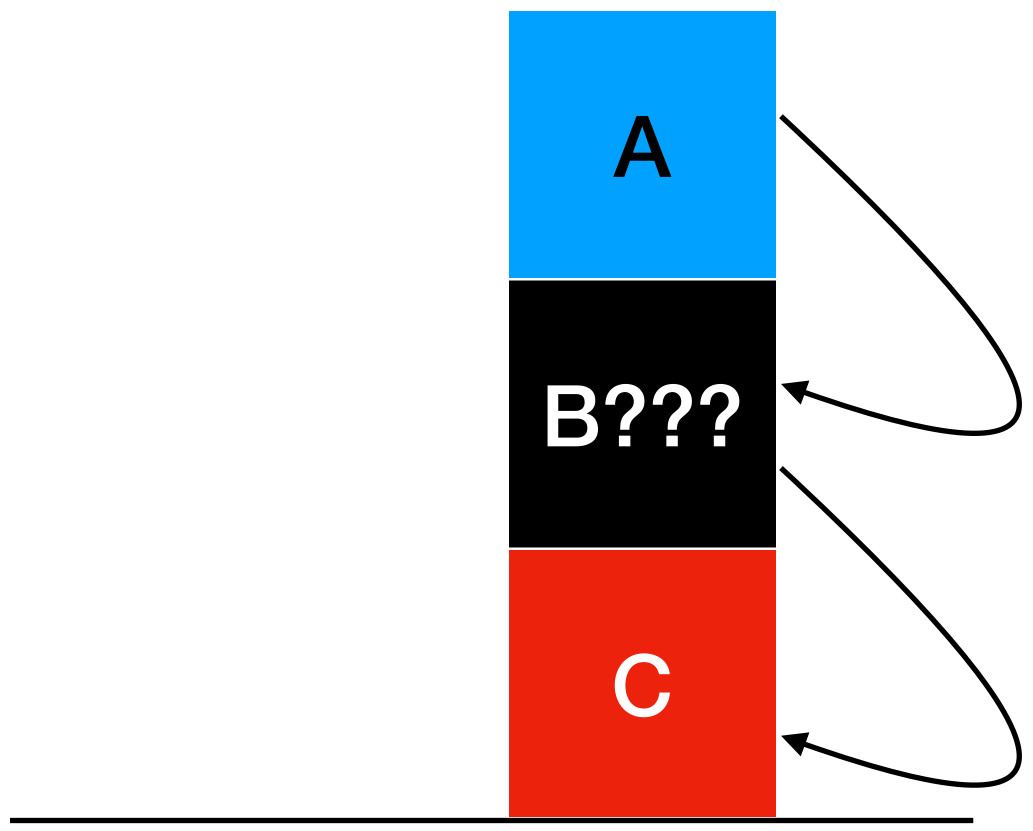


Some interpretations may not even make physical sense



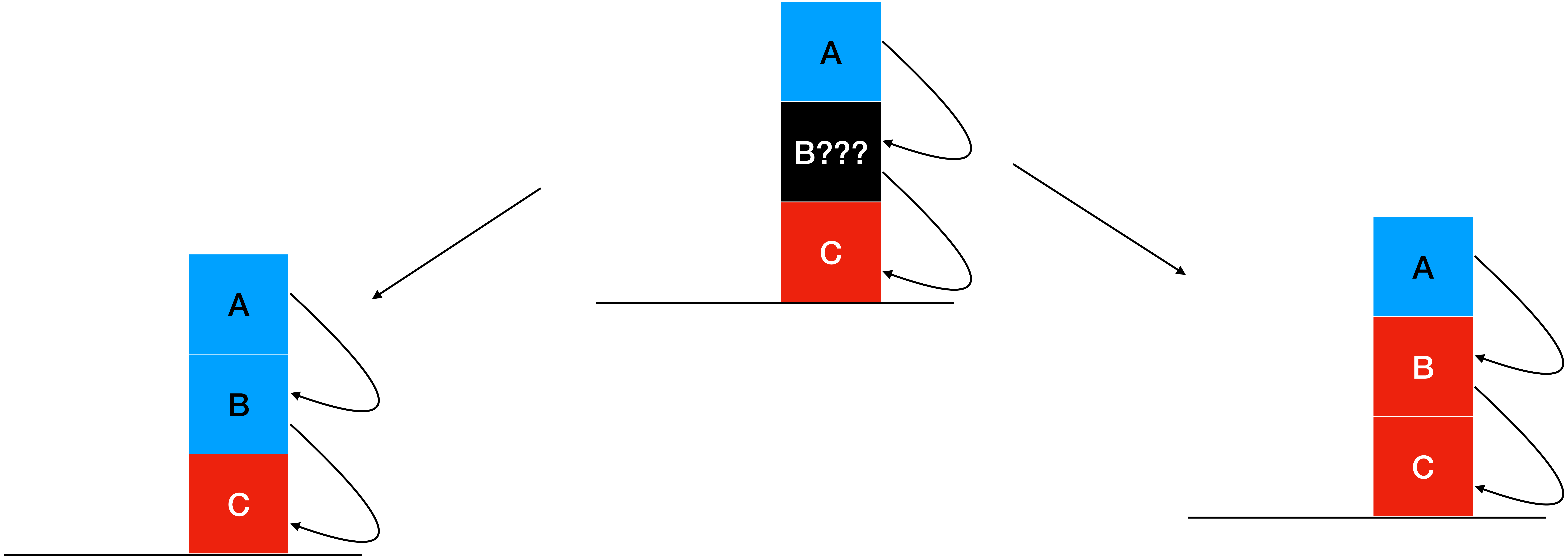
Example: block world

- What if the agent does not see the color of all blocks?
- Is there a blue block on top of a non blue block?



Example: block world

- Only 2 interpretations satisfy what the agent knows
- In both of them, there is a blue block on top of a non-blue, so yes



Deductive inference

- It's easy to explain the conclusion from the previous slide in English
- We want to do it mechanically
- Idea:
  - Express agent knowledge as symbols in a KB
  - Express the goal as the symbols of a "query" sentence α
  - Manipulate the KB symbols until we produce the symbols for "α is true" or "α is false"

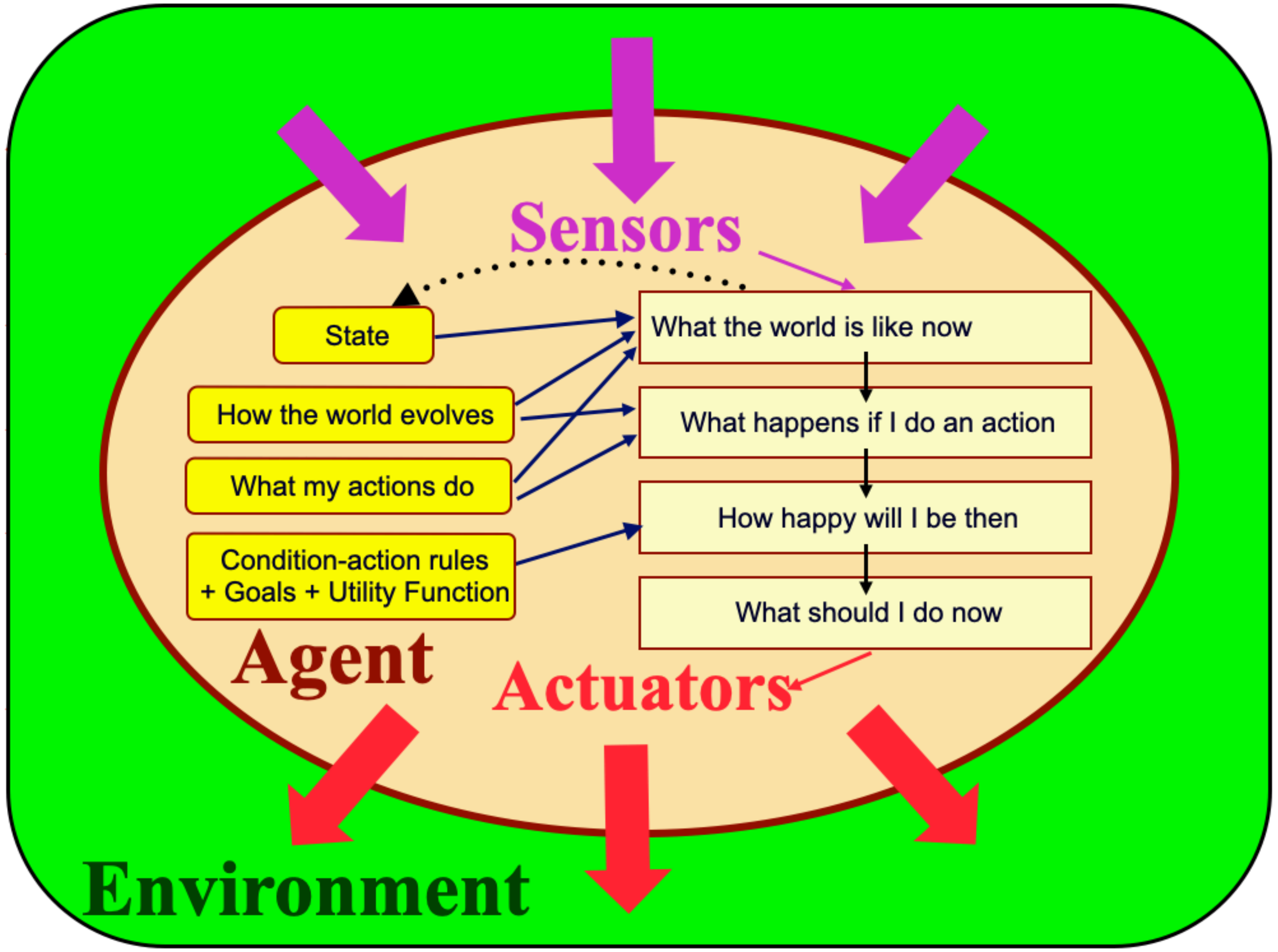
Deductive inference

- If a given algorithm i succeeds in proving α from the KB, we say it derives α
    - KB $\vdash_i \alpha$
- We want KB $\vdash_i \alpha$ if and only if KB $\models \alpha$
    - If an algorithm only derives sentences that are actually entailed, we say it is **sound**
    - If it always produces an answer, we say it is **complete**
- Unfortunately, no sound and complete algorithm for checking the entailment of sentences in First-Order Logic can exist

A note on agents

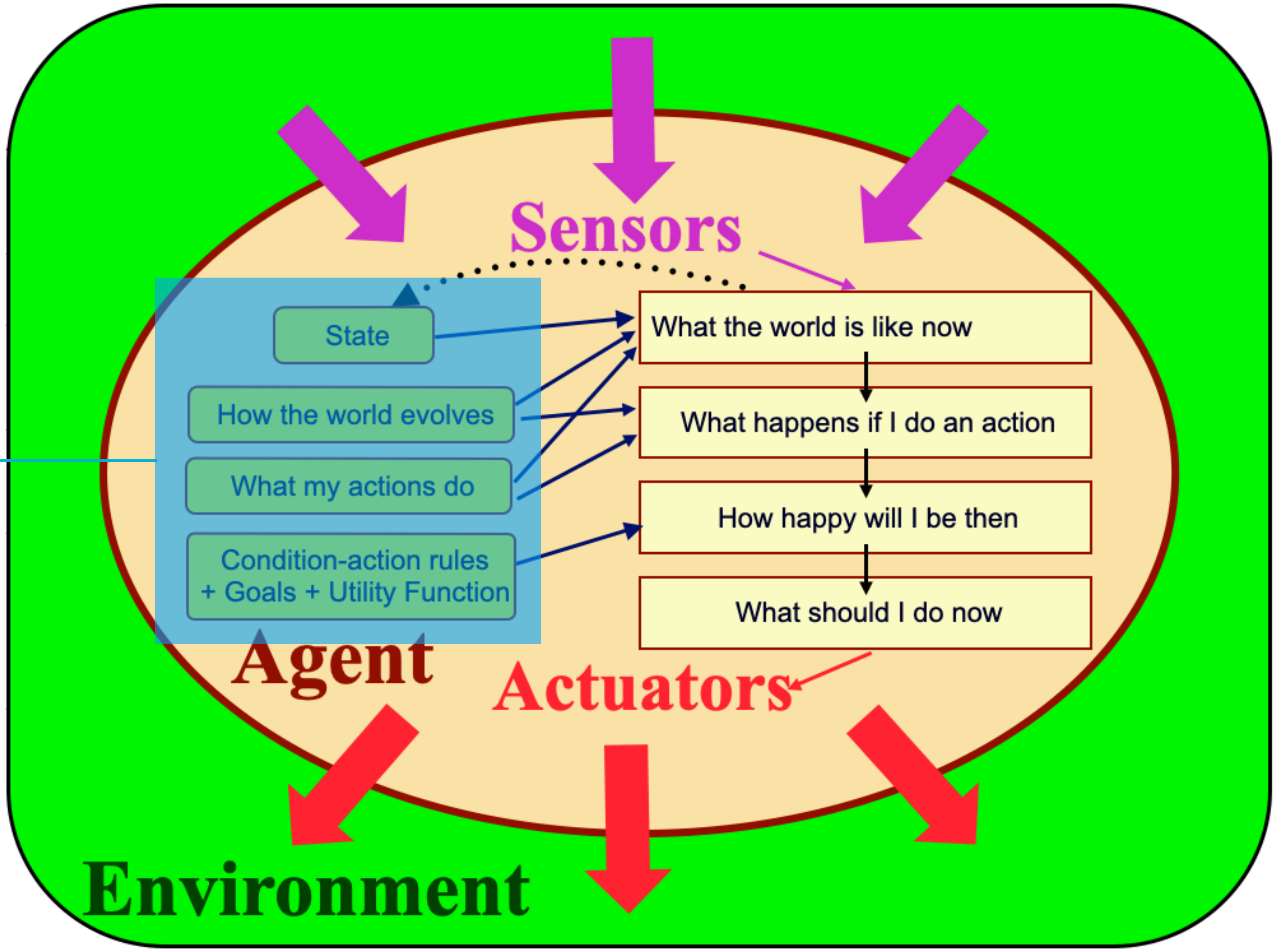
- Strictly speaking, logic tells us what knowledge can be inferred from a KB (that is, producing new knowledge from existing knowledge)
- However, we are often interested in *what is the right thing to do*.
- An agent is an entity (machine or human) that interacts with an environment trying to achieve a goal
- To bridge the gap between knowledge and action, we many want to check whether the KB entails propositions of the form “The correct thing to do is X”
- An agent that acts according to this principle is often called a knowledge-based agent

Example of Agent Diagram



Example of Agent Diagram

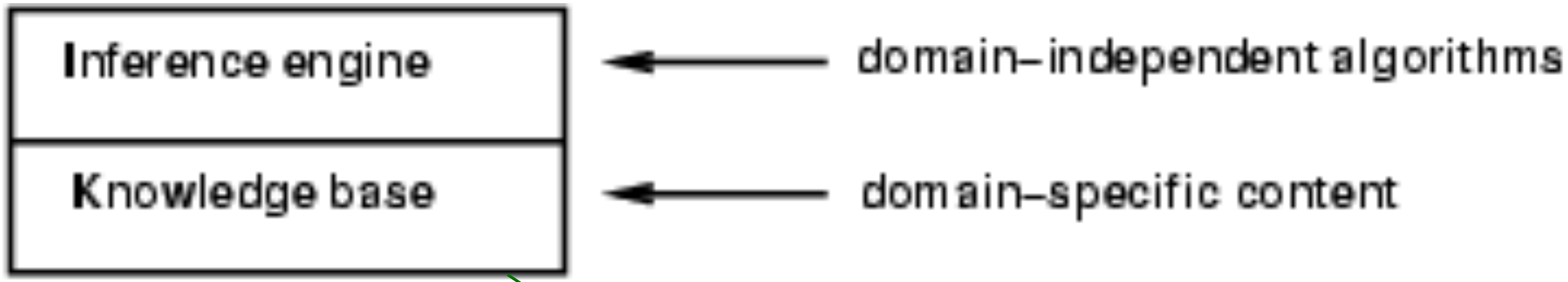
Knowledge Base



Knowledge-Based agents

Inference rules such as:

- A v B, ¬A ⊢ B



http://aima.eecs.berkeley.edu/slides-ppt/

“Facts” such as

- Fido is a dog
- There is a 2 in the top-left square

“Rules” such as

- Every dog is a mammal
- If two squares share a column, they must have different numbers

Knowledge-Based agents pseudocode

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

http://aima.eecs.berkeley.edu/slides-ppt/

Knowledge-Based agents pseudocode

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

http://aima.eecs.berkeley.edu/slides-ppt/

Building a KB

- The AIMA book suggests the following steps for knowledge engineering projects (ch. 8.4.1):

1. Identify the questions
2. Assemble the relevant knowledge
3. Decide on a vocabulary (predicates, functions, constants)
4. Encode general knowledge about the domain
5. Encode a description of the project instance
6. Pose queries to the inference procedure and get answers
7. Debug and evaluate the KB

Next class

- We will apply those steps to some more interesting problems
- Start reading chapter 3 of the book, entry ticket will be due Thursday
- I will assign a quiz based on chapter 2 in the next few days.