Knowledge and Action

Christoph Schwering

COMP4418, Week 8

Background

John McCarthy (1927–2011):

- Stanford, MIT, Dartmouth
- Turing Award
- Invented Lisp (1958)
- Invented Garbage Collection (1959)
- Founding Father of AI (with Minsky, Newell, Simon, 1955)

Background

John McCarthy (1927–2011):

- Stanford, MIT, Dartmouth
- Turing Award
- Invented Lisp (1958)
- Invented Garbage Collection (1959)
- Founding Father of AI (with Minsky, Newell, Simon, 1955)
- Proposed *Advice Taker* (1959)
  - *Programs with Common Sense*
  - Improve program behaviour by making statements to it
  - Program draws conclusions from its knowledge
    - Declarative conclusion: new **knowledge**
    - Imperative conclusion: take **action**
  - Remains a vision to this date

Advice Taker motivates (directly or indirectly) a lot of AI research to this date

Outline

Outline

Motivation

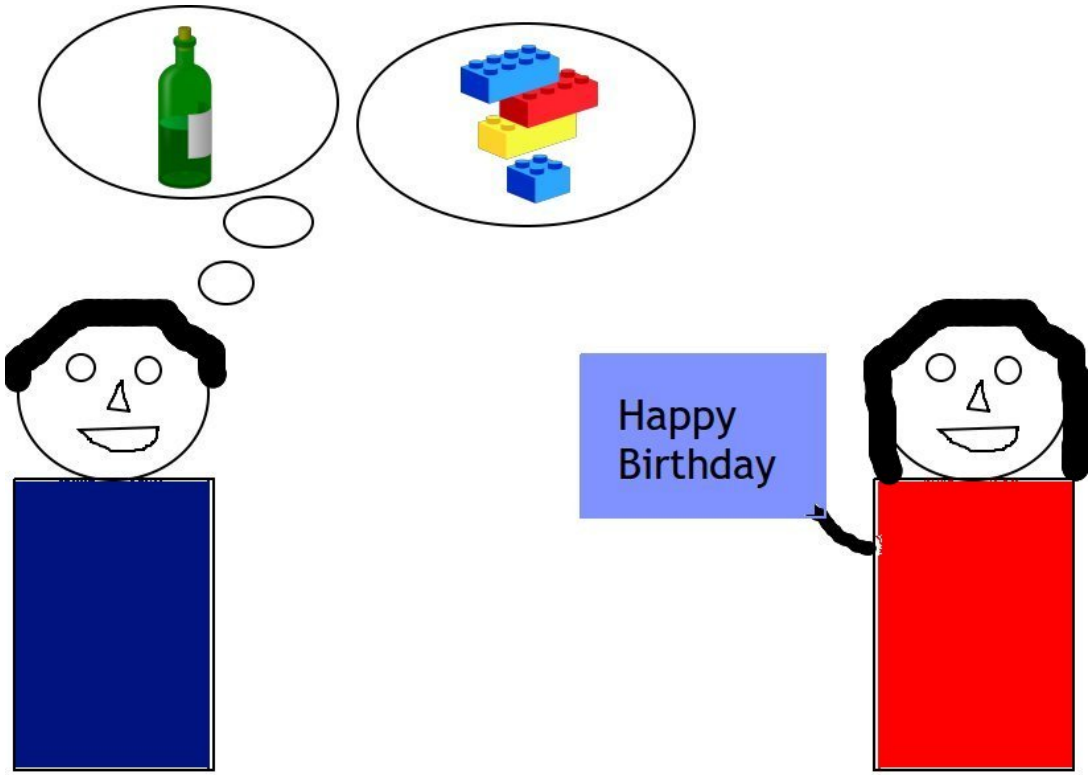
Observation: Non-knowledge is important

Not only what we *know* is relevant, but also what we *don't know*.

Motivation

Observation: Non-knowledge is important

Not only what we *know* is relevant, but also what we *don't know*.



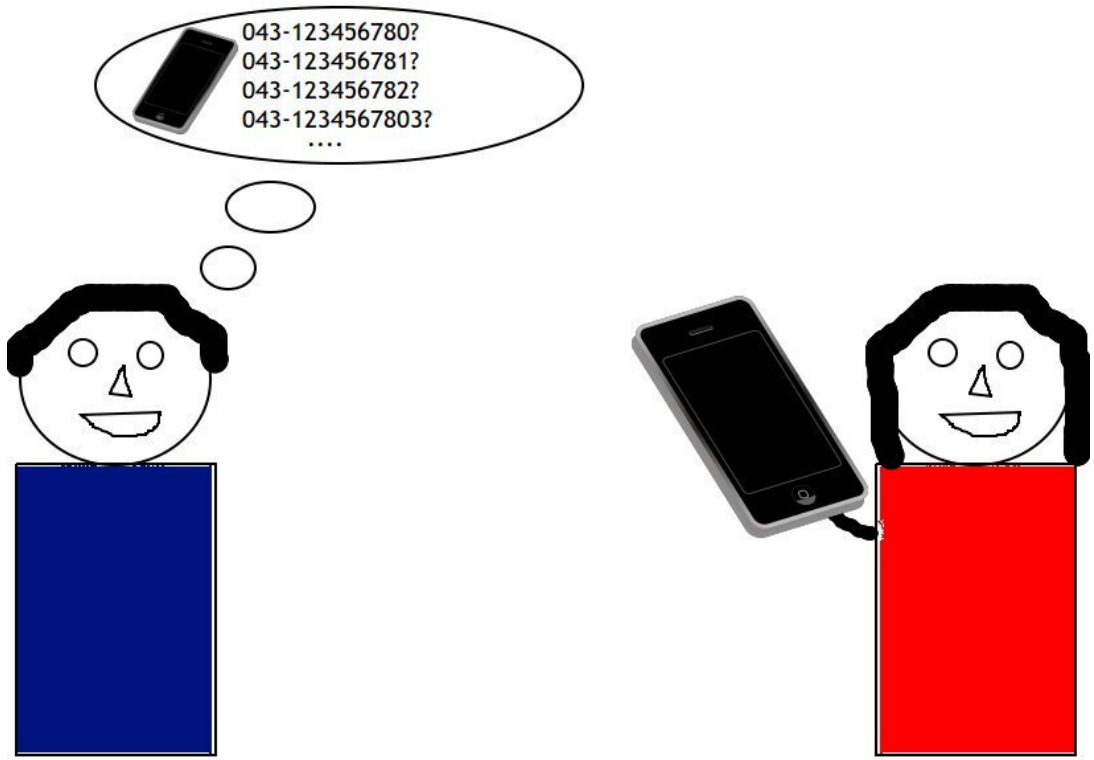
You don't know what's in the gift box.

So you'll treat it with great care.

Motivation

Observation: Non-knowledge is important

Not only what we *know* is relevant, but also what we *don't know*.



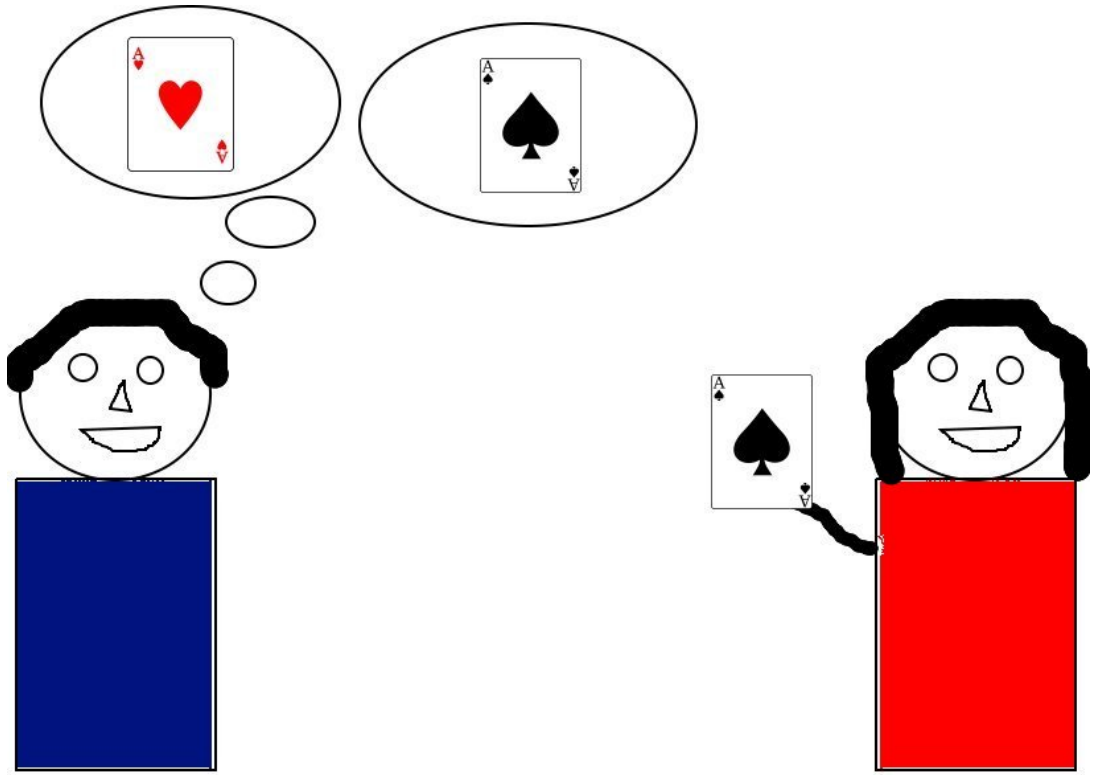
You know Jane has a phone, but you don't know her number.

So you'll look it up.

Motivation

Observation: Non-knowledge is important

Not only what we *know* is relevant, but also what we *don't know*.



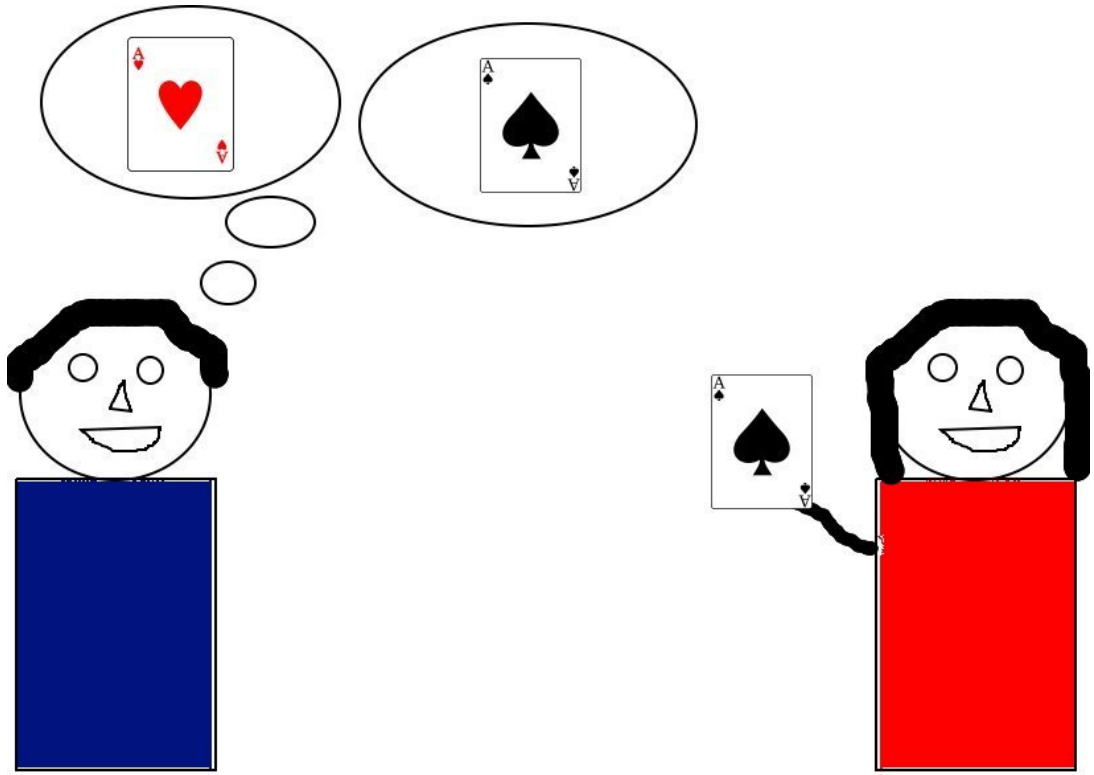
You know Jane is holding ace of spades *or* of hearts, but not which one.

So you'll look for a strategy that wins in either case.

Motivation

Observation: Non-knowledge is important

Not only what we *know* is relevant, but also what we *don't know*.



You know Jane is holding ace of spades *or* of hearts, but not which one.

So you'll look for a strategy that wins in either case.

How can we accurately **model knowledge and non-knowledge**?

Propositional Logic with Knowledge: Syntax

Syntax:

- Atomic propositions
- Negation: $\neg\phi$
- Disjunction: $(\phi_1 \vee \phi_2)$
- Conjunction: $(\phi_1 \wedge \phi_2)$
- Knowledge: $\mathbf{K}\phi$
- TRUE, FALSE, $\rightarrow$, $\leftrightarrow$ can be expressed with $\neg$, $\vee$, $\wedge$.

Examples:

1. $\mathbf{K}(p \vee \neg p)$
2. $\mathbf{K}(p \vee q) \rightarrow (\mathbf{K}p \vee \mathbf{K}q)$
3. $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
4. $p \rightarrow \mathbf{K}p$
5. $(\mathbf{K}p \vee \mathbf{K}q) \rightarrow \mathbf{K}(p \vee q)$

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: it is known that $(p \vee q)$.
  - $\neg\mathbf{K}p$: it is not known that p.
  - $\neg\mathbf{K}q$: it is not known that q.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: it is not known that p.
  - $\neg\mathbf{K}q$: it is not known that q.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: not all possible worlds satisfy p.
  - $\neg\mathbf{K}q$: it is not known that q.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world falsifies p.
  - $\neg\mathbf{K}q$: it is not known that q.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world satisfies $\neg p$.
  - $\neg\mathbf{K}q$: it is not known that q.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world satisfies $\neg p$.
  - $\neg\mathbf{K}q$: some possible world satisfies $\neg q$.

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a possibility what the real world could be.

- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.

- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible.
  - Knowledge is what is true in all possible worlds.

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world satisfies $\neg p$.
  - $\neg\mathbf{K}q$: some possible world satisfies $\neg q$.

  So there are at least two possible worlds: $\{p, \neg q\}$ and $\{\neg p, q\}$.

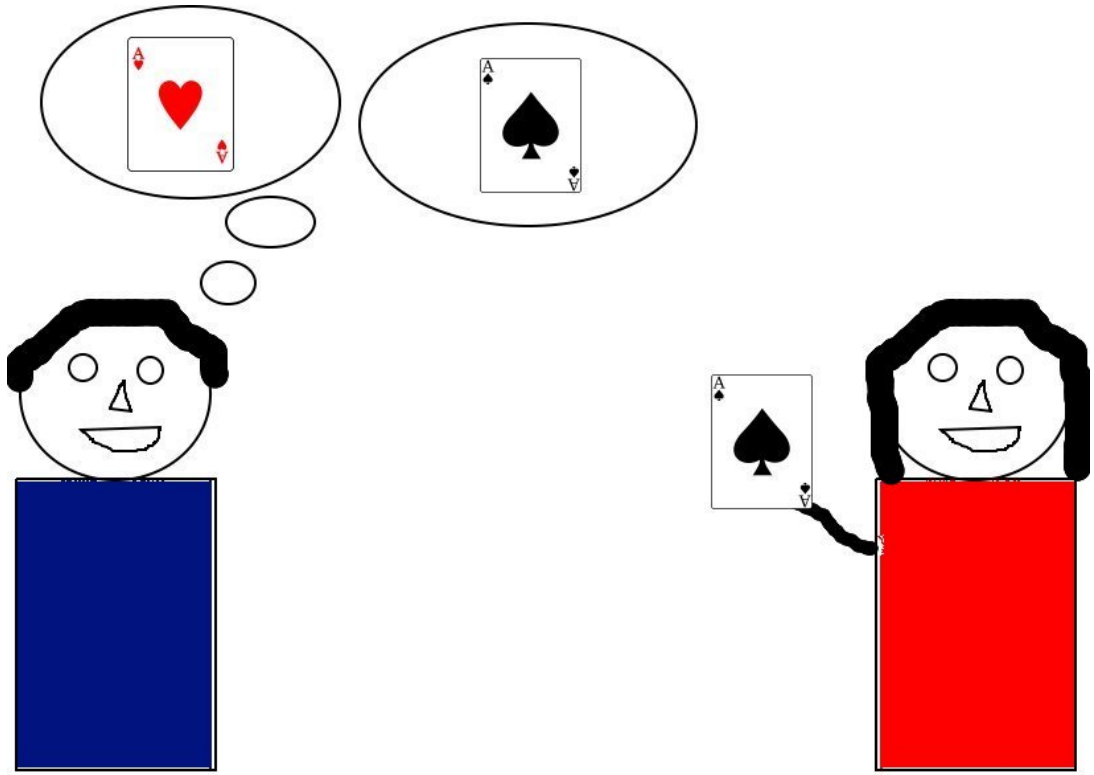
Possible-Worlds Semantics

Semantics:

- A world is a complete and consistent set of propositions.
- An interpretation is a pair e, w of a set of possible worlds e and a real world w.
- e, w satisfies p iff $p \in w$
- e, w satisfies $\neg\phi$ iff e, w does not satisfy ϕ
- e, w satisfies $(\phi_1 \vee \phi_2)$ iff e, w satisfies ϕ_1 or ϕ_2
- e, w satisfies $(\phi_1 \wedge \phi_2)$ iff e, w satisfies ϕ_1 and ϕ_2
- e, w satisfies $\mathbf{K}\phi$ iff for all $w' \in e$: e, w' satisfies ϕ

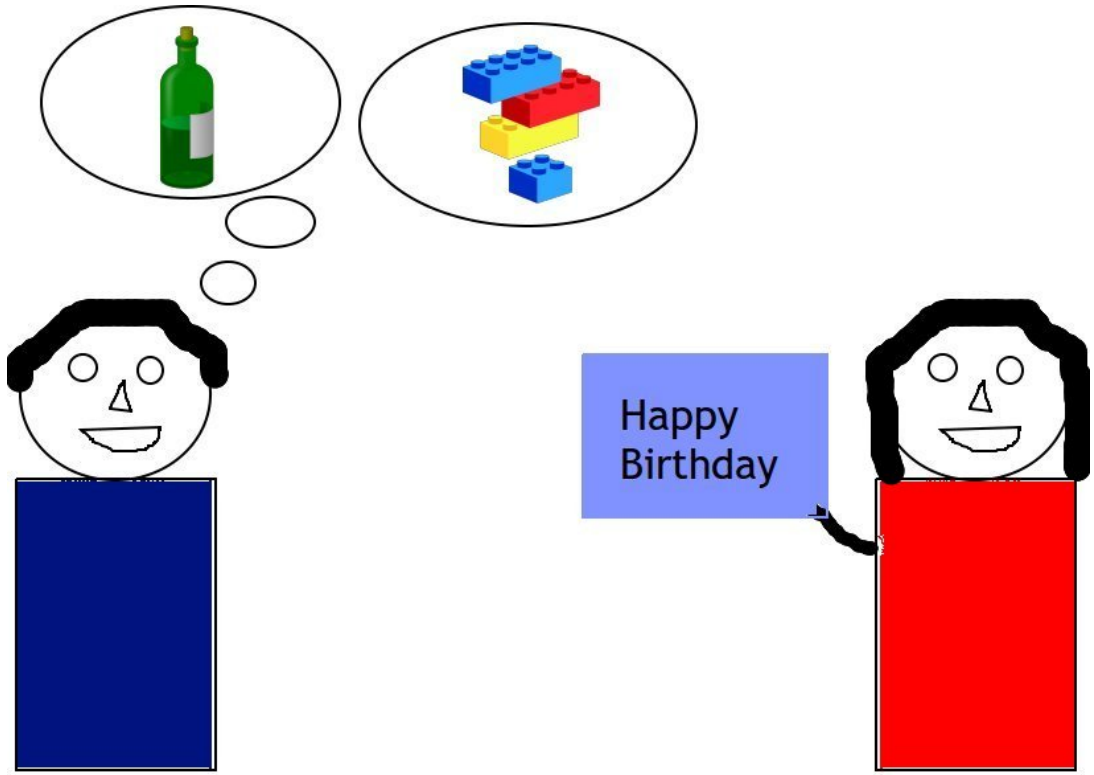
A more general characterisation uses Kripke structures.

Why is Propositional Logic Not Enough?



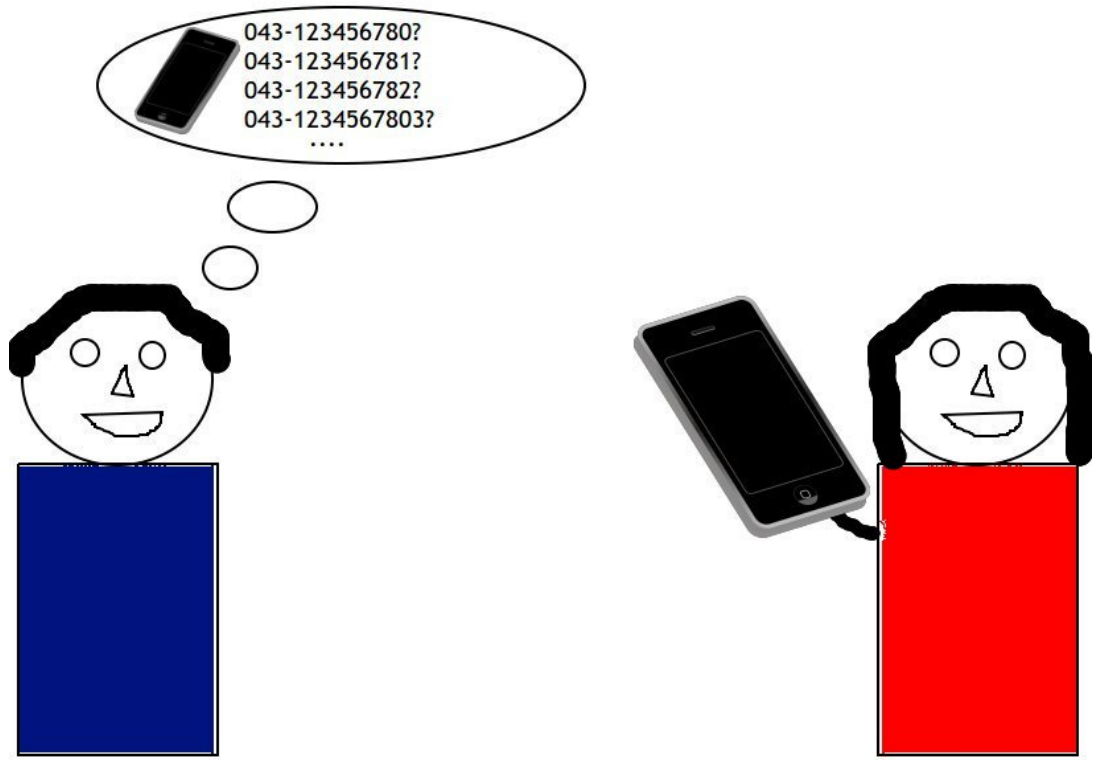
$$\mathbf{K}\big((\spadesuit \vee \heartsuit) \wedge \neg\mathbf{K}\spadesuit \wedge \neg\mathbf{K}\heartsuit\big)$$

Why is Propositional Logic Not Enough?



$$\mathbf{K}\exists x \left(\mathrm{InBox}(x) \wedge \neg \mathbf{K}\mathrm{InBox}(x)\right)$$

Why is Propositional Logic Not Enough?



$$\mathbf{K}\exists x\,(\mathrm{NumberOf}(\mathrm{Jane},x)\wedge\neg\mathbf{K}\mathrm{NumberOf}(\mathrm{Jane},x))$$

First-Order Logic with Knowledge: Syntax

Syntax:

- Terms: variables x, functions $f(t_1, \ldots, t_k)$
- Equality: $t_1 = t_2$
- Predicates: $P(t_1, \ldots, t_k)$
- Existential quantification: $\exists x \, \phi$
- Universal quantification: $\forall x \, \phi$

First-Order Logic with Knowledge: Syntax

Syntax:

- Terms: variables x, functions $f(t_1, \ldots, t_k)$
- Equality: $t_1 = t_2$
- Predicates: $P(t_1, \ldots, t_k)$
- Existential quantification: $\exists x \, \phi$
- Universal quantification: $\forall x \, \phi$
- Negation: $\neg \phi$
- Disjunction: $(\phi_1 \vee \phi_2)$
- Conjunction: $(\phi_1 \wedge \phi_2)$
- Knowledge: $\mathbf{K} \phi$
- $\neq$, TRUE, FALSE, $\rightarrow$, $\leftrightarrow$ can be expressed with $=$, $\neg$, $\vee$, $\wedge$.

Simplifications over Classical First-Order Logic

In classical first-order logic, the interpretation determines the domain and meaning of functions:

$$I = \langle D, \Phi \rangle$$

where

- D is a non-empty set, the *universe* or *domain*,
- Φ maps predicate symbols to relations over D, and
- Φ maps function symbols to functions over D.

In the Logic of Knowledge, we simplify things:

- We assume infinitely many function symbols of each arity.
- The domain contains all (ground) terms that can be formed from these function symbols.
- The interpretation of a function term is just the term itself: $\mathrm{fatherOf}(\mathrm{Sally})$ and $\mathrm{motherOf}(\mathrm{Sally})$ and Frank are all distinct.

First-Order Logic with Knowledge: Semantics

Semantics:

- A world is a complete and consistent set of ground predicates.
- An interpretation is a pair e, w of a set of possible worlds e and a real world w.
- e, w satisfies $t_1 = t_2$ iff t_1 and t_2 are identical
- e, w satisfies $P(t_1, \ldots, t_k)$ iff $P(t_1, \ldots, t_k) \in w$
- e, w satisfies $\exists x \phi$ iff for some ground term t: e, w satisfies ϕ_t^x
- e, w satisfies $\forall x \phi$ iff for all ground terms t: e, w satisfies ϕ_t^x

First-Order Logic with Knowledge: Semantics

Semantics:

- A world is a complete and consistent set of ground predicates.
- An interpretation is a pair e, w of a set of possible worlds e and a real world w.
- e, w satisfies $t_1 = t_2$ iff t_1 and t_2 are identical
- e, w satisfies $P(t_1, \ldots, t_k)$ iff $P(t_1, \ldots, t_k) \in w$
- e, w satisfies $\exists x \phi$ iff for some ground term t: e, w satisfies ϕ_t^x
- e, w satisfies $\forall x \phi$ iff for all ground terms t: e, w satisfies ϕ_t^x
- e, w satisfies $\neg\phi$ iff e, w does not satisfy ϕ
- e, w satisfies $(\phi_1 \vee \phi_2)$ iff e, w satisfies ϕ_1 or ϕ_2
- e, w satisfies $(\phi_1 \wedge \phi_2)$ iff e, w satisfies ϕ_1 and ϕ_2
- e, w satisfies $\mathbf{K}\phi$ iff for all $w' \in e$: e, w' satisfies ϕ

Multiple Agents

Syntax:

- Let A be an arbitrary set (think of it as set of agents).
- Knowledge for $a \in A$: $\mathbf{K}_a \phi$

Multiple Agents

Syntax:

- Let A be an arbitrary set (think of it as set of agents).
- Knowledge for $a \in A$: $\mathbf{K}_a \phi$

Semantics:

- Suppose $a, b \in A$.
  a considers some worlds possible.
  For each world, a considers some worlds to be possible from b's perspective.

Multiple Agents

Syntax:

- Let A be an arbitrary set (think of it as set of agents).
- Knowledge for $a \in A$: $\mathbf{K}_a \phi$

Semantics:

- Suppose $a, b \in A$.
  a considers some worlds possible.
  For each world, a considers some worlds to be possible from b's perspective.
- Let f be a function that maps each agent a and possible world w to a pair (e, f')
  - a set of possible worlds e and
  - a function f' of the same sort such that $f'(a, w') = (e, f')$ for all w'.
- f, w satisfies $\mathbf{K}_a \phi$ iff for $f(a, w) = (e, f')$:
  for all $w' \in e$: f', w' satisfies ϕ

`http://hintikkasworld.irisa.fr/`

Outline

Logical Omniscience

- An agent knows what is logically entailed by its knowledge base.
  - Let KB be the agent's knowledge base.
  - Then ϕ is known iff KB entails ϕ.

  The agent is logically omniscient.

- Complexity of "is ϕ known?":
  - Classical propositional logic: co-NP-complete
  - Classical first-order logic: undecidable

- Classical logic is an inappropriate model of human knowledge:
  - Complexity of classical logic is beyond human capabilities.
  - Most of the time, humans only use fragments of classical logic.

How to Control the Complexity?

- Satisfiability is NP-complete in propositional logic.
- Validity is co-NP-complete in propositional logic.
- Satisfiability/validity is undecidable in first-order logic.

Recall: ϕ is satisfiable iff $\neg\phi$ is not valid

ϕ is valid iff $\neg\phi$ is not satisfiable

How to Control the Complexity?

- Satisfiability is NP-complete in propositional logic.
- Validity is co-NP-complete in propositional logic.
- Satisfiability/validity is undecidable in first-order logic.

Recall: ϕ is satisfiable iff $\neg\phi$ is not valid
ϕ is valid iff $\neg\phi$ is not satisfiable

There are two ways to improve the complexity:

1. Restrict the language:
   - Only consider formulas of a certain form.
   - Satisfiability/validity of this subset has better complexity.

2. Restrict the semantics:
   - Weaken the entailment relation.
   - Soundness: $\models \phi$ implies $\mid\!\approx \phi$ usually wanted
   - Completeness: $\mid\!\approx \phi$ implies $\models \phi$ usually given up

Tractable Satisfiability in Classical Propositional Logic (1)

Definition: Horn logic

Horn logic: propositional CNF formulas with at most one positive literal per clause.

Tractable Satisfiability in Classical Propositional Logic (1)

Definition: Horn logic

Horn logic: propositional CNF formulas with at most one positive literal per clause.

Theorem: satisfiability for Horn logic

Satisfiability for Horn logic can be solved in linear time.

Tractable Satisfiability in Classical Propositional Logic (1)

Definition: Horn logic

Horn logic: propositional CNF formulas with at most one positive literal per clause.

Theorem: satisfiability for Horn logic

Satisfiability for Horn logic can be solved in linear time.

- Input: Horn formula ϕ of length n.
- Let I be the set of all unit clauses in ϕ.
- Loop until a fixpoint of I is reached:
  - If $(x_1 \vee \ldots \vee x_k) \in \phi$ and $\overline{x}_1, \ldots, \overline{x}_k \in I$: unsatisfiable.
  - If $(x_1 \vee \ldots \vee x_{k+1}) \in \phi$ and $\overline{x}_1, \ldots, \overline{x}_k \in I$: add x_{k+1} to I.
- Each clause needs to be traversed only once.
- This algorithm can be implemented with time complexity $\mathcal{O}(n)$.

Tractable Satisfiability in Classical Propositional Logic (2)

Definition: 2-CNF

2-CNF: propositional CNF formulas with two literals per clause.

Tractable Satisfiability in Classical Propositional Logic (2)

Definition: 2-CNF

2-CNF: propositional CNF formulas with two literals per clause.

Theorem: satisfiability for 2-CNF

Satisfiability for 2-CNF can be solved in linear time.

Tractable Satisfiability in Classical Propositional Logic (2)

Definition: 2-CNF

2-CNF: propositional CNF formulas with two literals per clause.

Theorem: satisfiability for 2-CNF

Satisfiability for 2-CNF can be solved in linear time.

- Input: 2-CNF formula ϕ of length n.
- Let ϕ^* be the least set such that
  - $\phi \subseteq \phi^*$ and
  - if $(x \vee y),\ (\overline{x} \vee z) \in \phi^*$, then $(y \vee z) \in \phi^*$.
- ϕ is equivalent to ϕ^*.
- ϕ is unsatisfiable iff $(x \vee x), (\neg x \vee \neg x) \in \phi^*$ for some x.
- $|\{x \mid x \text{ literal in } \phi\}| \leq n$.
- $|\{(x \vee y, \overline{x} \vee z) \mid y, z \text{ literal in } \phi\}| \leq n^2$ for each literal x.
- Time complexity of this algorithm is $\mathcal{O}(n^3)$, best known is $\mathcal{O}(n)$.

Decidable Satisfiability in Classical First-Order Logic (1)

A first-order formula is in prenex form iff quantifiers only occur in front of a quantifier-free rest: $Q_1 x_1 \ldots Q x_k \phi$ for quantifier-free ϕ.

Decidable Satisfiability in Classical First-Order Logic (1)

A first-order formula is in prenex form iff quantifiers only occur in front of a quantifier-free rest: $Q_1 x_1 \ldots Q x_k \phi$ for quantifier-free ϕ.

Definition: Bernays-Schönfinkel class

Bernays-Schönfinkel class: first-order formulas in prenex form with quantifier prefix $\exists^* \forall^*$ without functions.

Ex.: $\exists x \exists y \forall z \, (P(x, y) \wedge \neg P(x, z))$.

Decidable Satisfiability in Classical First-Order Logic (1)

A first-order formula is in prenex form iff quantifiers only occur in front of a quantifier-free rest: $Q_1 x_1 \ldots Q x_k \phi$ for quantifier-free ϕ.

Definition: Bernays-Schönfinkel class

Bernays-Schönfinkel class: first-order formulas in prenex form with quantifier prefix $\exists^* \forall^*$ without functions.

Ex.: $\exists x \exists y \forall z \, (P(x, y) \wedge \neg P(x, z))$.

Theorem: satisfiability for Bernays-Schönfinkel class

Satisfiability for the Bernays-Schönfinkel class is decidable.

Decidable Satisfiability in Classical First-Order Logic (1)

A first-order formula is in prenex form iff quantifiers only occur in front of a quantifier-free rest: $Q_1 x_1 \ldots Q x_k \phi$ for quantifier-free ϕ.

Definition: Bernays-Schönfinkel class

Bernays-Schönfinkel class: first-order formulas in prenex form with quantifier prefix $\exists^* \forall^*$ without functions.

Ex.: $\exists x \exists y \forall z \, (P(x, y) \wedge \neg P(x, z))$.

Theorem: satisfiability for Bernays-Schönfinkel class

Satisfiability for the Bernays-Schönfinkel class is decidable.

- Input: Bernays-Schönfinkel formula $\exists x_1 \ldots \exists x_k \forall y_1 \ldots \forall y_\ell \, \phi$
- Testing structures with domains of size k suffices:
  - Suppose $I = \langle D, \Phi \rangle$ satisfies ϕ.
  - For some $d_1, \ldots, d_k \in D$ substituted for $x_1, \ldots, x_k$, I satisfies ϕ.
  - $\langle \{d_1, \ldots, d_k\}, \Phi' \rangle$, where Φ' is Φ restricted to $d_1, \ldots, d_k$, satisfies ϕ.

Decidable Satisfiability in Classical First-Order Logic (2)

Definition: prefix-vocabulary class

A vocabulary-prefix class is the set of first-order formulas characterised by the following parameters:

- quantifier prefix: which quantifier combinations are allowed?
- predicates: how many predicates of which arity are allowed?
- functions: how many functions of which arity are allowed?
- equality: is equality allowed?

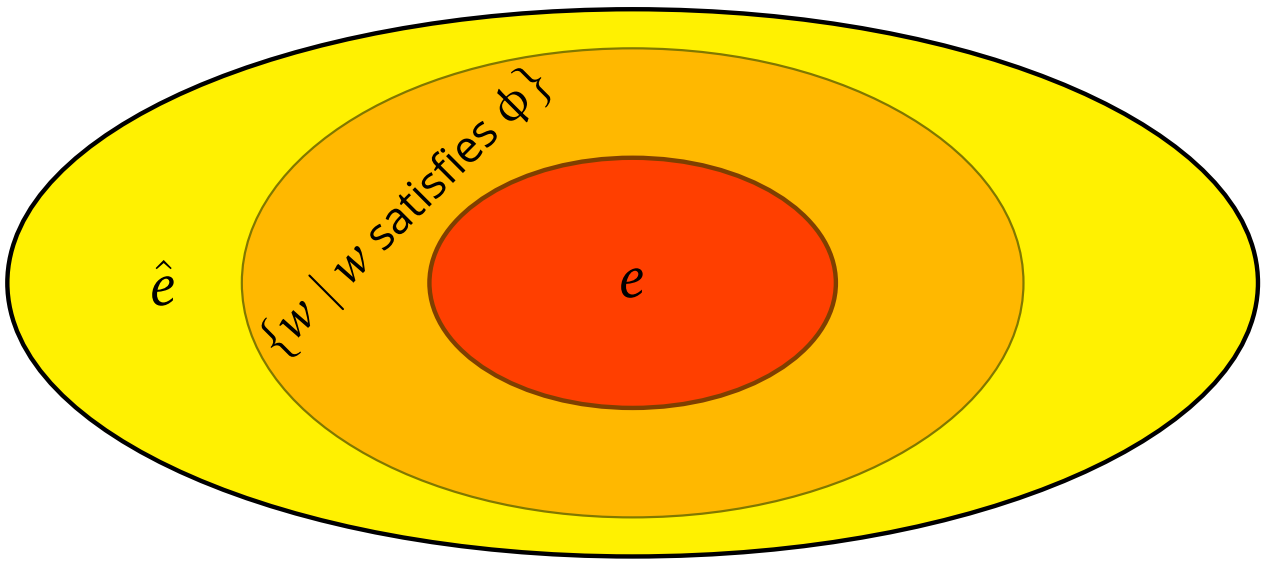
Quantifiers	Predicates	Functions	Equality
$\exists^*\forall^*$	all	none	yes
$\exists^*\forall\forall\exists^*$	all	none	no
all	all unary	all unary	no
$\exists^*\forall\exists^*$	all	all	no
$\exists^*$	all	all	yes
all	all unary	one unary	yes
$\exists^*\forall\exists^*$	all	one unary	yes

Avoiding Logical Omniscience (1)

Idea: Allow more possible worlds in e to know less.

Why?

- Larger e corresponds to less knowledge.
- e, w satisfies $\mathbf{K}\phi$ iff for all $w \in e$: e, w satisfies ϕ.
- For $\hat{e} \supseteq e$: $\hat{e} \models \mathbf{K}\phi$ implies $e \models \mathbf{K}\phi$.



Avoiding Logical Omniscience (2)

One cause of complexity of reasoning is closure under modus ponens:

> If ϕ is known and $(\phi \to \psi)$ is known, then ψ is known.

Why?

- $(\phi \to \psi)$ is equivalent to $(\neg\phi \vee \psi)$.
- If a world satisfies ϕ, it does not satisfy $\neg\phi$.
- If a world satisfies ϕ and $(\neg\phi \vee \psi)$, it satisfies ψ.

We can suppress modus ponens by allowing worlds to "satisfy", or support, ϕ and $\neg\phi$ at the same time.

Avoiding Logical Omniscience (3)

- A world is a (perhaps inconsistent) set of propositions.
- w supports p iff $p \in w$
- w supports $\neg p$ iff $\neg p \in w$
- w supports $\neg\neg\phi$ iff w supports ϕ
- w supports $(\phi_1 \vee \phi_2)$ iff w supports ϕ_1 or ϕ_2
- w supports $\neg(\phi_1 \vee \phi_2)$ iff w supports $\neg\phi_1$ and $\neg\phi_2$
- w supports $(\phi_1 \wedge \phi_2)$ iff w supports ϕ_1 and ϕ_2
- w supports $\neg(\phi_1 \wedge \phi_2)$ iff w supports $\neg\phi_1$ or $\neg\phi_2$
- e satisfies $\mathbf{K}\phi$ iff for all $w \in e$: w supports ϕ

Then $\mathbf{K}(p \wedge (\neg p \vee q))$ does not entail $\mathbf{K}q$:
$w = \{p, \neg p, \neg q\}$ supports p and $(\neg p \vee q)$, but not q.

Outline

Actions and Change: Three Problems

Three commonsense problems are fundamental to actions and change:

1. The Qualification Problem
2. The Frame Problem
3. The Ramification Problem

They are surprisingly difficult to solve.

The Qualification Problem

An action can only be executed under certain circumstances.

The Qualification Problem

Represent the preconditions (qualifications) of an action.

Ex.: You want to take a bus b to get to a destination d.

What must be true for this to be possible?

- Some qualifications are more important than others
  - Important qualification: d is on b's route
  - Minor qualification: fuel, driver, keys, ...
- Impractical to list all minor preconditions
- Non-monotonic reasoning
  - Action is possible when all important qualifications hold, unless a minor qualification prevents it
  - Not specific to actions: a bird flies unless it's abnormal

The Frame Problem

Most things do not change when an action is executed.

The Frame Problem

Represent what is left unchanged by an action (frame axioms).

Ex.: You don't magically disappear from the bus when it moves.

The weather also remains unchanged when the bus moves.

- Frame axioms specify what does *not* change
  - If you are on a bus, then you're still on the bus when it moves.
  - If you are not on a bus, then you're still not on the bus when it moves.

- m actions, n predicates $\implies$ about $2 \cdot m \cdot n$ frame axioms
  - 100 actions, 100 predicates $\implies$ 20 000 frame axioms
  - Impractical to write down
  - Need to generate them or represent them implicitly

State Constraints

State constraints must be satisfied over the course of actions.

The Ramification Problem

Represent indirect effects caused by state constraints.

Ex.: If you're on the bus, your location is where the bus is.

You cannot be at two busses at once.

- Indirect effect: action effects must adhere to state constraints
- Indirect qualification: action allowed only if state constraint won't be violated
- Constraints can often be compiled to qualifications, effects
  - When a bus moves, its passengers move along
  - You can get on a bus only if you're not on a bus already

Our Approach

We'll focus on the **frame problem**.

The Frame Problem

Represent what is left unchanged by an action.

Want: a way to *generate frame axioms* from given effect axioms.

Why?

- Modularity: this makes it easy to add new predicates / actions
- Accuracy: avoids the danger of forgetting frame axioms

First-Order Logic with Actions: Syntax

Syntax:

- Terms: variables x, functions $f(t_1, \ldots, t_k)$
- Equality: $t_1 = t_2$
- Predicates: $P(t_1, \ldots, t_k)$
- Single Actions: $[a]\phi$
- Action Sequence: $\Box\,\phi$

First-Order Logic with Actions: Syntax

Syntax:

- Terms: variables x, functions $f(t_1, \ldots, t_k)$
- Equality: $t_1 = t_2$
- Predicates: $P(t_1, \ldots, t_k)$
- Single Actions: $[a]\phi$
- Action Sequence: $\Box\,\phi$
- Existential quantification: $\exists x\,\phi$
- Universal quantification: $\forall x\,\phi$
- Negation: $\neg\phi$
- Disjunction: $(\phi_1 \vee \phi_2)$
- Conjunction: $(\phi_1 \wedge \phi_2)$
- Knowledge: $\mathbf{K}\phi$
- $\neq$, TRUE, FALSE, $\rightarrow$, $\leftrightarrow$ can be expressed with $=$, $\neg$, $\vee$, $\wedge$.

We defer the semantics.

Examples

- You don't fall off the bus when the bus moves:
  $\Box \forall b_1 \forall b_2 \forall d \left(\mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)] \mathrm{On}(b_1) \right)$
- You cannot be on two busses at once:
  $\Box \forall b_1 \forall b_2 \left(b_1 \neq b_2 \rightarrow \neg \mathrm{On}(b_1) \vee \neg \mathrm{On}(b_2) \right)$

Solving the Frame Problem with Successor-State Axioms

When are we on a bus?

Solving the Frame Problem with Successor-State Axioms

When are we on a bus?

What are the effects?

$\Box \forall b \, [\text{getOn}(b)]\text{On}(b)$

$\Box \forall b \, [\text{getOff}]\neg\text{On}(b)$

Solving the Frame Problem with Successor-State Axioms

When are we on a bus?

What are the effects?

$$\Box \forall a \forall b \left(a = \text{getOn}(b) \rightarrow [a]\text{On}(b) \right)$$

$$\Box \forall a \forall b \left(a = \text{getOff} \rightarrow [a]\neg\text{On}(b) \right)$$

Solving the Frame Problem with Successor-State Axioms

When are we on a bus?

What are the effects?

$$\Box \forall a \forall b \left(a = \text{getOn}(b) \rightarrow [a]\text{On}(b) \right)$$

$$\Box \forall a \forall b \left(a = \text{getOff} \rightarrow [a]\neg\text{On}(b) \right)$$

Assume causal completeness:

$$\Box \forall a \forall b \left(\neg\text{On}(b) \wedge [a] \;\; \text{On}(b) \rightarrow a = \text{getOn}(b) \right)$$

$$\Box \forall a \forall b \left(\;\; \text{On}(b) \wedge [a]\neg\text{On}(b) \rightarrow a = \text{getOff} \right)$$

Solving the Frame Problem with Successor-State Axioms

When are we on a bus?

What are the effects?

$$\Box \forall a \forall b \left(a = \text{getOn}(b) \rightarrow [a]\text{On}(b) \right)$$

$$\Box \forall a \forall b \left(a = \text{getOff} \rightarrow [a]\neg\text{On}(b) \right)$$

Assume causal completeness:

$$\Box \forall a \forall b \left(\neg\text{On}(b) \wedge [a] \;\; \text{On}(b) \rightarrow a = \text{getOn}(b) \right)$$

$$\Box \forall a \forall b \left(\;\; \text{On}(b) \wedge [a]\neg\text{On}(b) \rightarrow a = \text{getOff} \right)$$

These axioms are equivalent to a single successor-state axiom:

$$\Box \forall a \forall b \left([a]\text{On}(b) \leftrightarrow a = \text{getOn}(b) \vee (\text{On}(b) \wedge a \neq \text{getOff}) \right)$$

What's true *after* a is fully determined by what's true *before* a.

Successor-State Axioms

Definition: successor-state axiom

A **successor-state axiom** has the form

$$\Box \forall a \forall x_1 \ldots \forall x_k \left([a]P(x_1, \ldots, x_k) \leftrightarrow \gamma_P \right)$$

where γ_P does not mention $\Box$ or $[A]$ operators.

Typical form of γ_P is $\gamma_P^+ \vee (P(x_1, \ldots, x_k) \wedge \neg \gamma_P^-)$:

- γ_P^+ is the positive effect condition
- γ_P^- is the negative effect condition

Examples

You can get on and off a bus:

Positive effect: $\Box \forall a \forall b \, (a = \mathrm{getOn}(b) \rightarrow [a]\mathrm{On}(b))$

Negative effect: $\Box \forall a \forall b \, (a = \mathrm{getOff} \rightarrow [a]\neg\mathrm{On}(b))$

Examples

You can get on and off a bus:

Positive effect: $\Box \forall a \forall b \, (a = \mathrm{getOn}(b) \rightarrow [a]\mathrm{On}(b))$

Negative effect: $\Box \forall a \forall b \, (a = \mathrm{getOff} \rightarrow [a]\neg \mathrm{On}(b))$

SSA: $\Box \forall a \forall b \left([a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee (\mathrm{On}(b) \wedge \neg a = \mathrm{getOff}) \right)$

"You're on a bus iff you got on it *or*
you were on it and didn't get off.

Examples

You can change position using a bus:

Positive effect: $\Box \forall a \forall p \, (\exists b \, (a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b)) \rightarrow [a]\mathrm{At}(p))$

Negative effect: $\Box \forall a \forall b \, (\exists b \exists d \, (a = \mathrm{goTo}(b, d) \wedge d \neq p \wedge \mathrm{On}(b)) \rightarrow [a]\neg\mathrm{At}(p))$

Examples

You can change position using a bus:

Positive effect: $\Box \forall a \forall p \, (\exists b \, (a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b)) \rightarrow [a]\mathrm{At}(p))$

Negative effect: $\Box \forall a \forall b \, (\exists b \exists d \, (a = \mathrm{goTo}(b, d) \wedge d \neq p \wedge \mathrm{On}(b)) \rightarrow [a]\neg\mathrm{At}(p))$

SSA: $\Box \forall a \forall p \, \big([a]\mathrm{At}(p) \leftrightarrow \exists b \, \big(a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b)\big) \vee \big(\mathrm{At}(p) \wedge \neg\exists b \exists d \, (a = \mathrm{goTo}(b, d) \wedge d \neq p \wedge \mathrm{On}(b))\big)\big)$

"You're at a position p iff you were on a bus that moved to p *or* you were at p already and not on a bus that moved."

Projection

Definition: projection

Projection is the following decision problem:
Input: formulas ϕ, ψ, successor-state axioms Σ, actions $A_1, \ldots, A_\ell$.
Problem: do ϕ and Σ entail ψ after $A_1, \ldots, A_\ell$?

$$\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi?$$

Ex.: $\text{At}(\text{Central}) \wedge \Sigma \models [\text{getOn}(\text{M50})][\text{goTo}(\text{M50}, \text{Uni})]\text{At}(\text{Uni})?$

Projection

Definition: projection

Projection is the following decision problem:
Input: formulas ϕ, ψ, successor-state axioms Σ, actions $A_1, \ldots, A_\ell$.
Problem: do ϕ and Σ entail ψ after $A_1, \ldots, A_\ell$?

$$\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi?$$

Ex.: $\mathrm{At}(\mathrm{Central}) \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})?$

There are two approaches to projection problems:

- Regression: reduce to $\phi \models \psi^*$
- Progression: reduce to $\phi^* \models \psi$

Both reduce projection to ordinary entailment without actions.

Regression (1)

- Successor state axioms relate truth after a to truth before a:
$$\Box \forall a \forall x_1 \ldots \forall x_k \left([a]P(x_1, \ldots, x_k) \leftrightarrow \gamma_P \right)$$

- Let's define a procedure $\mathcal{R}(\phi)$ which iteratively replaces $[A]P(t_1, \ldots, t_k)$ with $\gamma_P \begin{smallmatrix} a & x_1 \ldots x_k \\ A & t_1 \ldots t_k \end{smallmatrix}$.

- $\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi$ iff $\phi \models \mathcal{R}([A_1] \ldots [A_\ell]\psi)$.

Regression (2)

- $\mathcal{R}([A_1] \dots [A_k](\alpha \vee \beta)) = (\mathcal{R}([A_1] \dots [A_k]\alpha) \vee \mathcal{R}([A_1] \dots [A_k]\beta))$

Regression (2)

- $\mathcal{R}([A_1] \dots [A_k](\alpha \vee \beta)) = (\mathcal{R}([A_1] \dots [A_k]\alpha) \vee \mathcal{R}([A_1] \dots [A_k]\beta))$
- $\mathcal{R}([A_1] \dots [A_k](\alpha \wedge \beta)) = (\mathcal{R}([A_1] \dots [A_k]\alpha) \wedge \mathcal{R}([A_1] \dots [A_k]\beta))$
- $\mathcal{R}([A_1] \dots [A_k]\neg\alpha) = \neg\mathcal{R}([A_1] \dots [A_k]\alpha)$
- $\mathcal{R}([A_1] \dots [A_k]\exists x\, \alpha) = \exists x\, \mathcal{R}([A_1] \dots [A_k]\alpha)$
- $\mathcal{R}([A_1] \dots [A_k]\forall x\, \alpha) = \forall x\, \mathcal{R}([A_1] \dots [A_k]\alpha)$
- $\mathcal{R}([A_1] \dots [A_k]t_1 = t_2) = t_1 = t_2$

Regression (2)

- $\mathcal{R}([A_1] \ldots [A_k](\alpha \vee \beta)) = (\mathcal{R}([A_1] \ldots [A_k]\alpha) \vee \mathcal{R}([A_1] \ldots [A_k]\beta))$
- $\mathcal{R}([A_1] \ldots [A_k](\alpha \wedge \beta)) = (\mathcal{R}([A_1] \ldots [A_k]\alpha) \wedge \mathcal{R}([A_1] \ldots [A_k]\beta))$
- $\mathcal{R}([A_1] \ldots [A_k]\neg\alpha) = \neg\mathcal{R}([A_1] \ldots [A_k]\alpha)$
- $\mathcal{R}([A_1] \ldots [A_k]\exists x\, \alpha) = \exists x\, \mathcal{R}([A_1] \ldots [A_k]\alpha)$
- $\mathcal{R}([A_1] \ldots [A_k]\forall x\, \alpha) = \forall x\, \mathcal{R}([A_1] \ldots [A_k]\alpha)$
- $\mathcal{R}([A_1] \ldots [A_k]t_1 = t_2) = t_1 = t_2$
- $\mathcal{R}(P(t_1, \ldots, t_k)) = P(t_1, \ldots, t_k)$
- $\mathcal{R}([A_1] \ldots [A_{\ell+1}]P(t_1, \ldots, t_k)) = \mathcal{R}([A_1] \ldots [A_\ell]\gamma_P \begin{smallmatrix} a & x_1 \ldots x_k \\ A_{\ell+1} & t_1 \ldots t_k \end{smallmatrix})$
  where
  - γ_P is the right-hand side of the successor-state axiom of P and
  - variables in γ_P are renamed to avoid clashes with variables in t_i.

The Regression Result

Theorem: regression

Let ϕ, ψ be without $[A]$ and $\Box$.
Let $A_1, \ldots, A_\ell$ be ground terms.
Let Σ be a set of successor-state axioms.
Then:

$$\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi \text{ iff } \phi \models \mathcal{R}([A_1] \ldots [A_\ell]\psi)$$

Example

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$ **?**

Example

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $$\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$$

Example

$$\Box \forall a \forall p \left([a]\mathrm{At}(p) \leftrightarrow \exists b \left(a = \mathrm{goTo}(b,p) \wedge \mathrm{On}(b) \right) \vee \left(\mathrm{At}(p) \wedge \neg \exists b \exists d \left(a = \mathrm{goTo}(b,d) \wedge d \neq p \wedge \mathrm{On}(b) \right) \right) \right)$$

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

Example

$$\Box \forall a \forall p \left([a]\mathrm{At}(p) \leftrightarrow \exists b \left(a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b) \right) \vee \left(\mathrm{At}(p) \wedge \neg \exists b \exists d \left(a = \mathrm{goTo}(b, d) \wedge d \neq p \wedge \mathrm{On}(b) \right) \right) \right)$$

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

Example

$$\Box \forall a \forall p \left([a]\mathrm{At}(p) \leftrightarrow \exists b \left(a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b) \right) \vee \left(\mathrm{At}(p) \wedge \neg \exists b \exists d \left(a = \mathrm{goTo}(b, d) \wedge d \neq p \wedge \mathrm{On}(b) \right) \right) \right)$$

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b) \right)) \vee \ldots$

Example

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b)\right))\vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\mathrm{On}(b))\right) \vee \ldots$

Example

$$\Box \forall a \forall b \left([a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee (\mathrm{On}(b) \wedge a \neq \mathrm{getOff}) \right)$$

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b) \right)) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\mathrm{On}(b)) \right) \vee \ldots$

Example

$$\Box \forall a \forall b \left([a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee (\mathrm{On}(b) \wedge a \neq \mathrm{getOff}) \right)$$

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b) \right)) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\mathrm{On}(b)) \right) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}(\gamma_{\mathrm{On}}{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b}) \right) \vee \ldots$

Example

$$\Box \forall a \forall b \left([a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee (\mathrm{On}(b) \wedge a \neq \mathrm{getOff}) \right)$$

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b)\right)) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\mathrm{On}(b))\right) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}(\gamma_{\mathrm{On}}{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b})\right) \vee \ldots$

iff $\phi \models \exists b \big(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge$
$\left(\mathrm{getOn}(\mathrm{M50}) = \mathrm{getOn}(b) \vee \right.$
$\left.\left(\mathcal{R}(\mathrm{On}(b)) \wedge \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff}\right)\right)\big) \vee \ldots$

Example

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b\,(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b)))) \vee \ldots$

iff $\phi \models \exists b\,(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\mathrm{On}(b))) \vee \ldots$

iff $\phi \models \exists b\,(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}(\gamma_{\mathrm{On}}{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b})) \vee \ldots$

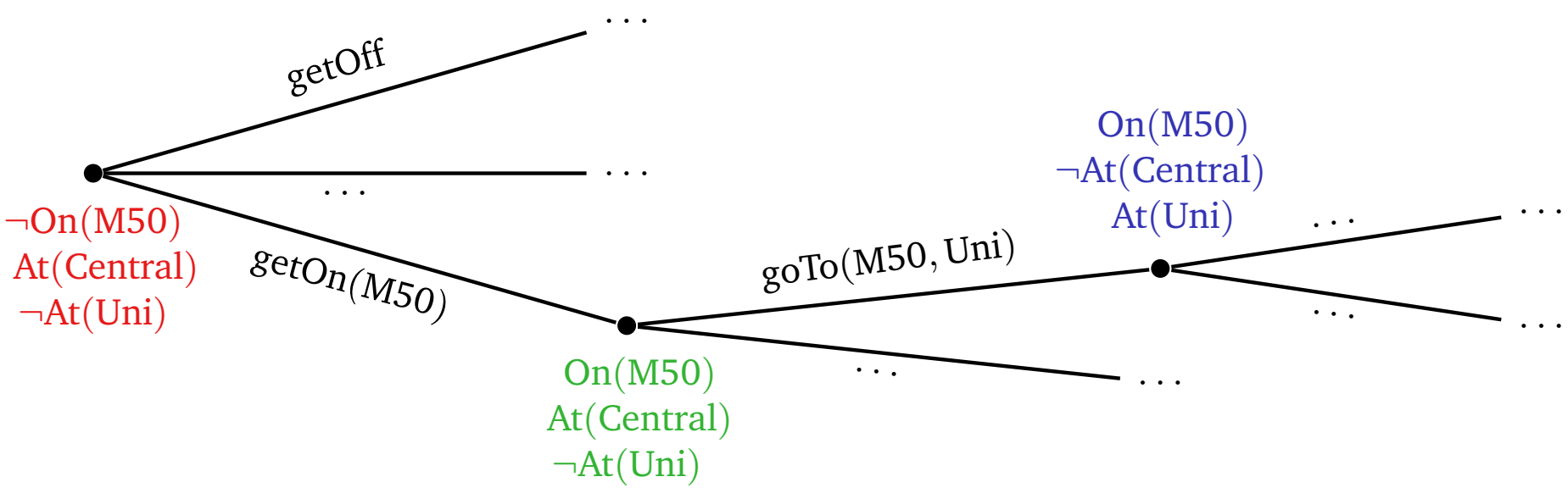
iff $\phi \models \exists b\,(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge$
$(\mathrm{getOn}(\mathrm{M50}) = \mathrm{getOn}(b) \vee$
$(\mathcal{R}(\mathrm{On}(b)) \wedge \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff}))) \vee \ldots$

iff $\phi \models \exists b\,\underbrace{(\mathrm{M50} = b \wedge (\mathrm{M50} = b \vee \mathcal{R}(\mathrm{On}(b))))}\vee \ldots$ ✓

Valid if b is M50. So the whole formula is valid and hence entailed by ϕ.

Worlds with Actions

A world is a tree of complete and consistent sets of literals:



$w \gg A$ denotes subtree with whose root is the node reached by A.

$$w \supseteq \{\neg \text{On(M50)}, \text{At(Central)}, \neg \text{At(Uni)}\}$$
$$w \gg \text{getOn(M50)} \supseteq \{\text{On(M50)}, \text{At(Central)}, \neg \text{At(Uni)}\}$$
$$w \gg \text{getOn(M50)} \gg \text{goTo(M50, Uni)} \supseteq \{\text{On(M50)}, \neg \text{At(Central)}, \text{At(Uni)}\}$$

First-Order Logic with Actions: Semantics

Semantics:

- A interpretation is a world w.
- w satisfies $P(t_1, \ldots, t_k)$ iff $P(t_1, \ldots, t_k) \in w$
- w satisfies $[A]\phi$ iff $w \gg A$ satisfies ϕ
- w satisfies $\Box\,\phi$ iff for all $A_1, \ldots, A_\ell$: $w \gg A_1 \gg \ldots \gg A_\ell$ satisfies ϕ

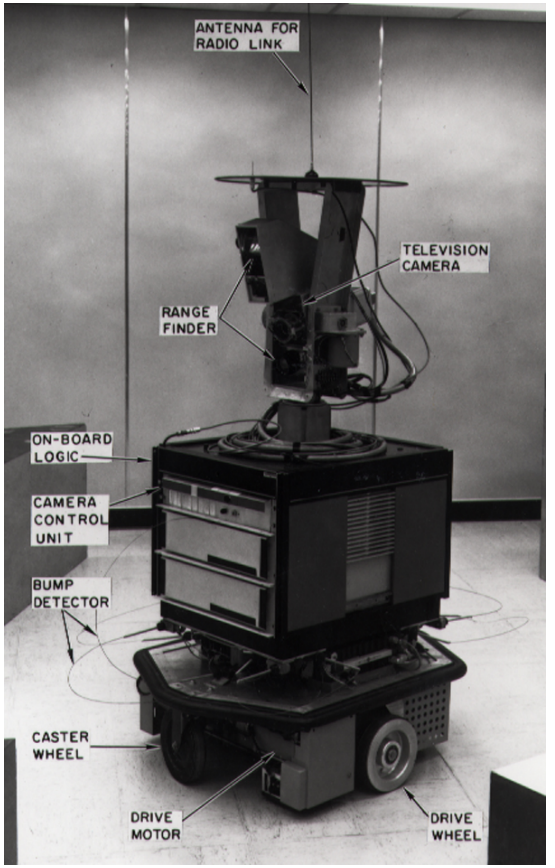
First-Order Logic with Actions: Semantics

Semantics:

- A interpretation is a world w.
- w satisfies $P(t_1, \ldots, t_k)$ iff $P(t_1, \ldots, t_k) \in w$
- w satisfies $[A]\phi$ iff $w \gg A$ satisfies ϕ
- w satisfies $\Box\,\phi$ iff for all $A_1, \ldots, A_\ell$: $w \gg A_1 \gg \ldots \gg A_\ell$ satisfies ϕ
- w satisfies $t_1 = t_2$ iff t_1 and t_2 are identical
- w satisfies $\neg\phi$ iff w does not satisfy ϕ
- w satisfies $(\phi_1 \vee \phi_2)$ iff w satisfies ϕ_1 or ϕ_2
- w satisfies $(\phi_1 \wedge \phi_2)$ iff w satisfies ϕ_1 and ϕ_2
- w satisfies $\exists x\,\phi$ iff for some ground term t: w satisfies ϕ^x_t
- w satisfies $\forall x\,\phi$ iff for all ground terms t: w satisfies ϕ^x_t

Outline

Planning: Background



- In the beginning (1950ies, 1960ies): reasoning about action = planning.
- McCarthy's logic of actions (1963, 1969) was too complex for practical purposes.
- Shakey introduced STRIPS planning (1971).
- Reasoning about action & change and planning diverged.
- They're converging:
  - Reasoning about action & change becomes more efficient.
  - Planning becomes more expressive.
  - Both sides benefit from each other.

STRIPS Planning

Stanford Research Institute Problem Solver

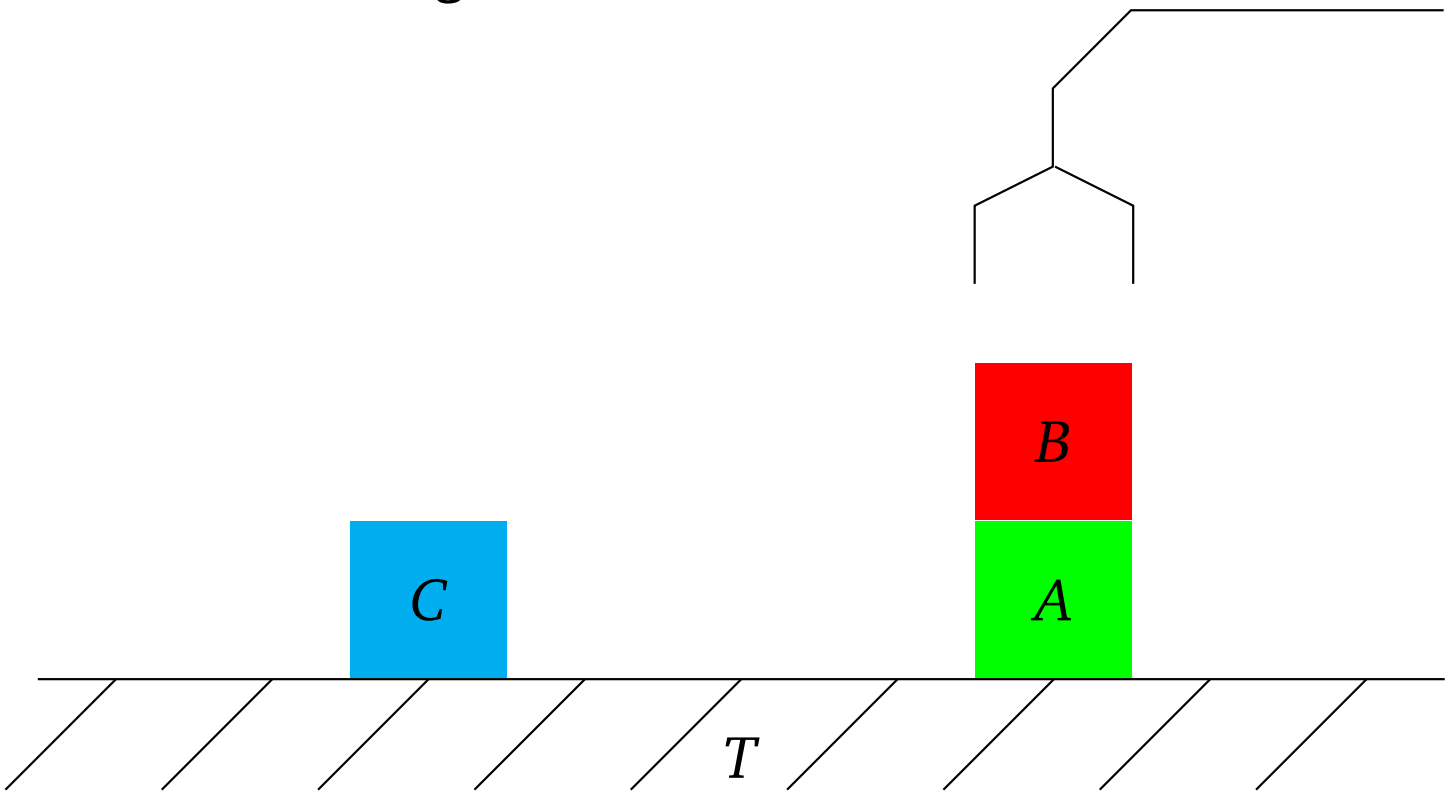
- A state is a set of propositions.
  - Propositions in the current state are true.
  - Propositions not in the current state are false.
- Action operators consist of:
  - Precondition: propositions that must be true
  - Positive effects: propositions that are added to state
  - Negative effects: propositions that are deleted from state

A problem instance consists of:

- Initial state
- Goal state
- Set of action operators

STRIPS Example: Blocks World (1)

A robot is stacking blocks on a table:



STRIPS Example: Blocks World (2)

- Initial state: $\text{Block}(A)$, $\text{Block}(B)$, $\text{Block}(C)$, HandEmpty, $\text{On}(C, T)$, $\text{Clear}(C)$, $\text{On}(A, T)$, $\text{On}(B, A)$, $\text{Clear}(B)$

- Action $\text{pickUp}(x, y)$ picks up block x when it is on y:
  - Precondition: $\text{Block}(x)$, $\text{On}(x, y)$, $\text{Clear}(x)$, HandEmpty
  - Positive effects: $\text{Holding}(x)$, $\text{Clear}(y)$
  - Negative effects: $\text{On}(x, y)$, $\text{Clear}(x)$, HandEmpty

- Action $\text{putOn}(x, y)$ puts block x on block y:
  - Precondition: $\text{Block}(x)$, $\text{Block}(y)$, $\text{Holding}(x)$, $\text{Clear}(y)$
  - Positive effects: $\text{On}(x, y)$, $\text{Clear}(x)$, HandEmpty
  - Negative effects: $\text{Holding}(x)$, $\text{Clear}(y)$

- Action $\text{putOnTable}(x)$ puts x on the table:
  - Precondition: $\text{Block}(x)$, $\text{Holding}(x)$
  - Positive effects: $\text{On}(x, T)$, $\text{Clear}(x)$, HandEmpty
  - Negative effects: $\text{Holding}(x)$

Limitations of STRIPS

The representation language of STRIPS is very restrictive:

- The world is assumed to be fully known.
  - Not expressible: it is unknown whether Scott is PM.
- Disjunctions cannot be expressed.
  - Not expressible: Scott is PM or Malcolm is PM.
- The “variables” are merely shorthands.
  - Not expressible: Someone is the PM.
- Actions cannot have conditional effects.
  - Not expressible: a switch toggles light on/off.
- Actions cannot have non-local effects.
  - Not expressivel: when a bus moves, the passengers move, too.

Some Questions

These are some questions you should be able to answer:

- Answer Set Programming
  - What are the differences to Prolog?
  - What is the rationality principle?
  - What is a normal logic rule/program?
  - What is a reduct, what is a stable model?
  - What are integrity constraints etc. and other extensions?
  - What is the complexity of ASP?
  - How are variables handled?
  - What is the typical structure of an ASP program?
- Satisfiability
- Knowledge
- Actions

Some Questions

These are some questions you should be able to answer:

- Answer Set Programming
- Satisfiability
  - How is SAT defined?
  - Why CNF, why not DNF?
  - What is the complexity of SAT and k-SAT?
  - Why is SAT so important?
  - How do Algorithms 1–3 work?
  - What is the invariant of the Watched-Literal Scheme, and why?
  - Why does Clause Learning help?
  - Which other ways of improving SAT solvers are there?
- Knowledge
- Actions

Some Questions

These are some questions you should be able to answer:

- Answer Set Programming
- Satisfiability
- Knowledge
  - What is the semantics of knowledge?
  - What is the difference between $\mathbf{K}p \vee \mathbf{K}q$ and $\mathbf{K}(p \vee q)$?
  - What is the difference between $\mathbf{K}\exists x P(x)$ and $\exists x \mathbf{K}P(x)$?
- Actions
  - What are the three main problems related to actions & change?
  - How does a successor-state axiom look like, and why?
  - What is projection?
  - How does regression work?
  - What is STRIPS and how does it relate to our logic?