# Reasoning about Actions

Christoph Schwering

UNSW Sydney

COMP4418, Week 9

# Reasoning about Actions

- McCarthy's Advice Taker
  - Improve program behaviour by making statements to it
  - Program draws conclusions from its knowledge
    - Declarative conclusion: new knowledge
    - Imperative conclusion: take **action**

# Reasoning about Actions

- McCarthy's Advice Taker
  - Improve program behaviour by making statements to it
  - Program draws conclusions from its knowledge
    - Declarative conclusion: new knowledge
    - Imperative conclusion: take **action**

- Actions change the environment, modify fluents
  - When you get on a bus, you are on the bus
  - When you get off a bus, you are not on the bus
  - When a bus moves, the position of the passengers changes

# Reasoning about Actions

- McCarthy's Advice Taker
  - Improve program behaviour by making statements to it
  - Program draws conclusions from its knowledge
    - Declarative conclusion: new knowledge
    - Imperative conclusion: take **action**

- Actions change the environment, modify fluents
  - When you get on a bus, you are on the bus
  - When you get off a bus, you are not on the bus
  - When a bus moves, the position of the passengers changes

- Want to model such environments
  - Action theory that models the actions and fluents
  - What does this theory entail?

# Overview of the Lecture

- **Three Problems**
- The Situation Calculus
- Projection by regression
- Projection by progression
- Knowledge and sensing
- Concluding words

# Three Problems

Commonsense problems, seemingly easy, yet very hard to formalise:

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

# The Qualification Problem

An action can only be executed under certain circumstances.

# The Qualification Problem

An action can only be executed under certain circumstances.

**The Qualification Problem**

Represent the preconditions (qualifications) of an action.

# 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?

# 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, . . .

# 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

# 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 fluents are not affected by an action.

# The Frame Problem

Most fluents are not affected by an action.

**The Frame Problem**

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

# The Frame Problem

Most fluents are not affected by an action.

**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.

# The Frame Problem

Most fluents are not affected by an action.

**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.

# The Frame Problem

Most fluents are not affected by an action.

**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.
- A actions, F fluents $\implies$ about $2 \times A \times F$ frame axioms
  - 100 actions, 100 fluents $\implies$ 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.

# State Constraints

State constraints must be satisfied over the course of actions.

**The Ramification Problem**

Represent indirect effects caused by state constraints.

# 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.

# 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

# 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

# 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 (due to Ray Reiter)

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

## The Frame Problem

Represent what is left unchanged by an action.

- Simple solution to the frame problem due to Reiter:
  $F$ holds after $a \iff a$ enables $F$ *or*
  $F$ holds before $a$ and $a$ does not disable $F$
- Ignore the minor qualifications
- Compile state constraints to qualifications and effects

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

- Modularity: could easily add new fluents / actions
- Accuracy: wouldn't forget frame axioms

# Overview of the Lecture

- Three Problems
- **The Situation Calculus**
- Projection by regression
- Projection by progression
- Knowledge and sensing
- Concluding words

# The Language of the Situation Calculus

**Terms** of two different **sorts**:

- Variables, standard names, functions of sort $\begin{cases}\text{object} \\ \text{action}\end{cases}$

# The Language of the Situation Calculus

**Terms** of two different **sorts**:

- Variables, standard names, functions of sort $\begin{cases} \text{object} \\ \text{action} \end{cases}$
- For simplicity: no nested functions, function only on left-hand side
- Special condition: action term $A(n_1, \ldots, n_j)$ is standard name

# The Language of the Situation Calculus

**Terms** of two different **sorts**:

- Variables, standard names, functions of sort $\begin{cases} \text{object} \\ \text{action} \end{cases}$
- For simplicity: no nested functions, function only on left-hand side
- Special condition: action term $A(n_1, \ldots, n_j)$ is standard name

Ex.: If $\mathrm{M50}$ is an object standard name and $\mathrm{getOn}$ is an action function, then $\mathrm{getOn}(\mathrm{M50})$ is an action standard name.

# The Language of the Situation Calculus

**Terms** of two different **sorts**:

- Variables, standard names, functions of sort $\begin{cases} \text{object} \\ \text{action} \end{cases}$
- For simplicity: no nested functions, function only on left-hand side
- Special condition: action term $A(n_1, \ldots, n_j)$ is standard name

Ex.: If $\mathrm{M50}$ is an object standard name and $\mathrm{getOn}$ is an action function, then $\mathrm{getOn}(\mathrm{M50})$ is an action standard name.
Then $\models \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff} \neq \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) \neq \ldots$!

# The Language of the Situation Calculus

**Terms** of two different **sorts**:

- Variables, standard names, functions of sort $\begin{cases}\text{object} \\ \text{action}\end{cases}$
- For simplicity: no nested functions, function only on left-hand side
- Special condition: action term $A(n_1, \ldots, n_j)$ is standard name

Ex.: If $\mathrm{M50}$ is an object standard name and $\mathrm{getOn}$ is an action function, then $\mathrm{getOn}(\mathrm{M50})$ is an action standard name.
Then $\models \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff} \neq \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) \neq \ldots$!

**Formulas**:

- $P(t_1, \ldots, t_j)$ $\quad t_1 = t_2$ $\quad \neg\alpha$ $\quad (\alpha \vee \beta)$ $\quad \exists x\, \alpha$
- $[t]\alpha$ — $\alpha$ holds after action $t$
- $\Box\, \alpha$ — $\alpha$ holds after any sequence of actions
- Predicate $\mathrm{Poss}(t)$ represents precondition of action $t$

# Examples and Convention

- You don't fall off the bus when the bus moves:
  $\Box\left(\forall b_1 \forall b_2 \forall d \left(\mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)]\mathrm{On}(b_1)\right)\right)$

- You cannot be on two busses at once:
  $\Box\left(\forall b_1 \forall b_2 \left(b_1 \neq b_2 \rightarrow \neg\mathrm{On}(b_1) \vee \neg\mathrm{On}(b_2)\right)\right)$

- $F$ holds after $a$ $\iff$ $a$ enables $F$ *or*
  $F$ holds before $a$ and $a$ does not disable $F$
  $\Box\left(\forall a \forall \vec{x}\left([a]F(\vec{x}) \leftrightarrow \gamma^+ \vee (F(\vec{x}) \wedge \neg\gamma^-)\right)\right)$

Convention:

- $\forall \vec{t}$ stands for $\forall t_1 \ldots \forall t_j$, $F(\vec{t})$ for $F(t_1, \ldots, t_j)$

# Examples and Convention

- You don't fall off the bus when the bus moves:
  $\Box\left(\forall b_1 \forall b_2 \forall d\left(\mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)]\mathrm{On}(b_1)\right)\right)$

- You cannot be on two busses at once:
  $\Box\left(\forall b_1 \forall b_2\left(b_1 \neq b_2 \rightarrow \neg\mathrm{On}(b_1) \vee \neg\mathrm{On}(b_2)\right)\right)$

- $F$ holds after $a$ $\iff$ $a$ enables $F$ *or*
  $F$ holds before $a$ and $a$ does not disable $F$
  $\Box\left(\forall a \forall \vec{x}\left([a]F(\vec{x}) \leftrightarrow \gamma^{+} \vee (F(\vec{x}) \wedge \neg\gamma^{-})\right)\right)$

Convention:

- $\forall\vec{t}$ stands for $\forall t_1 \ldots \forall t_j$, $F(\vec{t})$ for $F(t_1, \ldots, t_j)$
- Operator $\Box$ has maximum scope

# Examples and Convention

- You don't fall off the bus when the bus moves:
  $\Box\left(\forall b_1 \forall b_2 \forall d \left(\mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)]\mathrm{On}(b_1)\right)\right)$

- You cannot be on two busses at once:
  $\Box\left(\forall b_1 \forall b_2 \left(b_1 \neq b_2 \rightarrow \neg\mathrm{On}(b_1) \vee \neg\mathrm{On}(b_2)\right)\right)$

- $F$ holds after $a$ $\iff$ $a$ enables $F$ *or*
  $F$ holds before $a$ and $a$ does not disable $F$
  $\Box\left(\forall a \forall \vec{x} \left([a]F(\vec{x}) \leftrightarrow \gamma^{+} \vee (F(\vec{x}) \wedge \neg\gamma^{-})\right)\right)$

Convention:

- $\forall \vec{t}$ stands for $\forall t_1 \ldots \forall t_j$, $F(\vec{t})$ for $F(t_1, \ldots, t_j)$
- Operator $\Box$ has maximum scope
- Free variables are implicitly universally quantified

# Examples and Convention

- You don't fall off the bus when the bus moves:
  $\Box\, \mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)]\mathrm{On}(b_1)$

- You cannot be on two busses at once:
  $\Box\, b_1 \neq b_2 \rightarrow \neg\mathrm{On}(b_1) \vee \neg\mathrm{On}(b_2)$

- $F$ holds after $a$ $\iff$ $a$ enables $F$ *or*
  $F$ holds before $a$ and $a$ does not disable $F$
  $\Box\, [a]F(\vec{x}) \leftrightarrow \gamma^+ \vee (F(\vec{x}) \wedge \neg\gamma^-)$

Convention:

- $\forall\vec{t}$ stands for $\forall t_1 \ldots \forall t_j$, $F(\vec{t})$ for $F(t_1, \ldots, t_j)$
- Operator $\Box$ has maximum scope
- Free variables are implicitly universally quantified

# Examples and Convention

- You don't fall off the bus when the bus moves:
  $\Box\, \mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)]\mathrm{On}(b_1)$

- You cannot be on two busses at once:
  $\Box\, b_1 \neq b_2 \rightarrow \neg\mathrm{On}(b_1) \vee \neg\mathrm{On}(b_2)$

- $F$ holds after $a \iff a$ enables $F$ *or*
  $F$ holds before $a$ and $a$ does not disable $F$
  $\Box\, [a]F(\vec{x}) \leftrightarrow \gamma^{+} \vee (F(\vec{x}) \wedge \neg\gamma^{-})$

Convention:

- $\forall\vec{t}$ stands for $\forall t_1 \ldots \forall t_j$, $F(\vec{t})$ for $F(t_1, \ldots, t_j)$
- Operator $\Box$ has maximum scope
- Free variables are implicitly universally quantified
- We sometimes identify a (finite) set $\Sigma$ of sentences $\{\alpha_1, \ldots, \alpha_j\}$ with the conjunction $\alpha_1 \wedge \ldots \wedge \alpha_j$

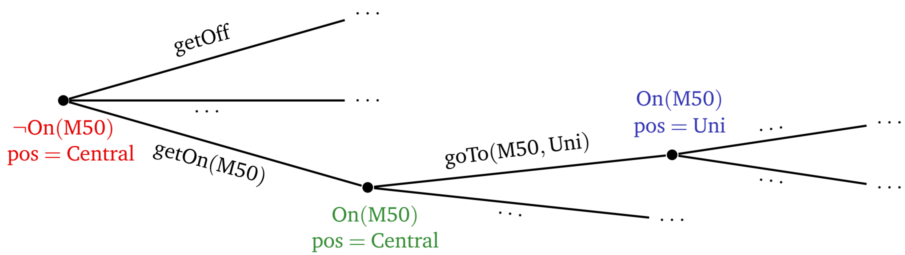
# Worlds and Situations

$$w[\mathrm{On}(\mathrm{M50}),\ \langle\rangle] = 0$$
$$w[\mathrm{pos},\ \langle\rangle] = \mathrm{Central}$$
$$w[\mathrm{On}(\mathrm{M50}),\ \mathrm{getOn}(\mathrm{M50})] = 1$$
$$w[\mathrm{pos},\ \mathrm{getOn}(\mathrm{M50})] = \mathrm{Central}$$
$$w[\mathrm{On}(\mathrm{M50}),\ \mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})] = 1$$
$$w[\mathrm{pos},\ \mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})] = \mathrm{Uni}$$

# Worlds and Situations

$$w[\mathrm{On}(\mathrm{M50}),\ \langle\rangle] = 0$$
$$w[\mathrm{pos},\ \langle\rangle] = \mathrm{Central}$$
$$w[\mathrm{On}(\mathrm{M50}),\ \mathrm{getOn}(\mathrm{M50})] = 1$$
$$w[\mathrm{pos},\ \mathrm{getOn}(\mathrm{M50})] = \mathrm{Central}$$
$$w[\mathrm{On}(\mathrm{M50}),\ \mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})] = 1$$
$$w[\mathrm{pos},\ \mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})] = \mathrm{Uni}$$

Tree view of $w$:



# Worlds and Situations (2)

**Definition: situation, world**

A **situation** $z$ is a sequence of action standard names.

# Worlds and Situations (2)

## Definition: situation, world

A **situation** $z$ is a sequence of action standard names.

A **world** $w$ is a function that maps

- primitive functions $f(\vec{n})$ and situations to standard names, and
- primitive atomic formulas $P(\vec{n})$ and situations to $\{0, 1\}$.

# Worlds and Situations (2)

## Definition: situation, world

A **situation** $z$ is a sequence of action standard names.

A **world** $w$ is a function that maps

- primitive functions $f(\vec{n})$ and situations to standard names, and
- primitive atomic formulas $P(\vec{n})$ and situations to $\{0, 1\}$.

The **denotation** of a ground term w.r.t. $w$ in $z$ is defined as

- $w(n, z) \stackrel{\text{def}}{=} n$ for every standard name $n$
- $w(f(n_1, \ldots, n_j), z) \stackrel{\text{def}}{=} w[f(n_1, \ldots, n_j), z]$

Recall: for simplicity we don't consider nested functions, so $f$ can only be applied to variables or names

# The Semantics of the Situation Calculus

## Definition: semantics

- $w, z \models P(t_1, \ldots, t_j) \iff w[P(w(t_1, z), \ldots, w(t_j, z), z] = 1$
- $w, z \models t_1 = t_2 \iff w(t_1, z) = w(t_2, z)$

# The Semantics of the Situation Calculus

## Definition: semantics

- $w, z \models P(t_1, \ldots, t_j) \iff w[P(w(t_1, z), \ldots, w(t_j, z), z] = 1$
- $w, z \models t_1 = t_2 \iff w(t_1, z) = w(t_2, z)$
- $w, z \models \neg\alpha \iff w, z \not\models \alpha$
- $w, z \models (\alpha \vee \beta) \iff w, z \models \alpha$ or $w, z \models \beta$
- $w, z \models \exists x\, \alpha \iff w, z \models \alpha^x_n$ for some std. name $n$ of $x$'s sort

# The Semantics of the Situation Calculus

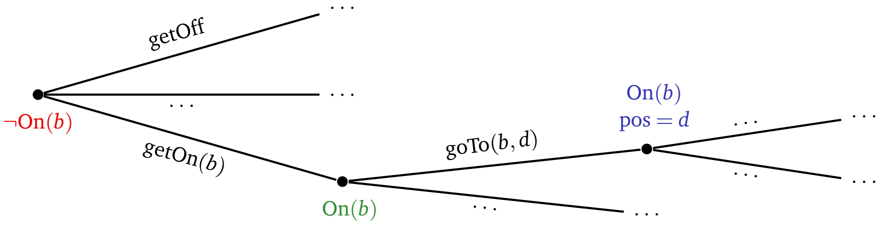
## Definition: semantics

- $w, z \models P(t_1, \ldots, t_j) \iff w[P(w(t_1, z), \ldots, w(t_j, z), z] = 1$
- $w, z \models t_1 = t_2 \iff w(t_1, z) = w(t_2, z)$
- $w, z \models \neg\alpha \iff w, z \not\models \alpha$
- $w, z \models (\alpha \vee \beta) \iff w, z \models \alpha$ or $w, z \models \beta$
- $w, z \models \exists x\, \alpha \iff w, z \models \alpha^x_n$ for some std. name $n$ of $x$'s sort
- $w, z \models [n]\alpha \iff w, z \cdot n \models \alpha$
- $w, z \models \Box\, \alpha \iff w, z \cdot z' \models \alpha$ for all situations $z'$

$\Sigma \models \alpha \iff$ for all $w$, if $w, \langle\rangle \models \beta$ for all $\beta \in \Sigma$, then $w, \langle\rangle \models \alpha$

# Example

$$w \models \neg \mathrm{On}(b)$$
$$w \models [\mathrm{getOn}(b)]\mathrm{On}(b)$$
$$w \models [\mathrm{getOn}(b)][\mathrm{goTo}(b,d)]\mathrm{On}(b)$$
$$w \models [\mathrm{getOn}(b)][\mathrm{goTo}(b,d)]\mathrm{pos} = d$$
$$w \models \exists a_1 \exists a_2 [a_1][a_2]\mathrm{pos} = d$$



# Solving the Frame Problem – Reiter's Idea

When are we on a bus?

# Solving the Frame Problem – Reiter's Idea

When are we on a bus?

Effect axioms:

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

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

# Solving the Frame Problem – Reiter's Idea

When are we on a bus?

Effect axioms:

$$\Box\, a = \mathrm{getOn}(b) \to [a]\mathrm{On}(b)$$

$$\Box\, a = \mathrm{getOff} \to [a]\neg\mathrm{On}(b)$$

# Solving the Frame Problem – Reiter's Idea

When are we on a bus?

Effect axioms:

$$\Box\, a = \text{getOn}(b) \to [a]\text{On}(b)$$

$$\Box\, a = \text{getOff} \to [a]\neg\text{On}(b)$$

Assume **causal completeness**, i.e., assume:

$$\Box\, \neg\text{On}(b) \wedge [a]\ \ \text{On}(b) \to a = \text{getOn}(b)$$

$$\Box\ \ \ \text{On}(b) \wedge [a]\neg\text{On}(b) \to a = \text{getOff}$$

# Solving the Frame Problem – Reiter's Idea

When are we on a bus?

Effect axioms:

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

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

Assume **causal completeness**, i.e., assume:

$$\Box\, \neg\mathrm{On}(b) \wedge [a]\ \ \mathrm{On}(b) \rightarrow a = \mathrm{getOn}(b)$$

$$\Box\ \ \ \mathrm{On}(b) \wedge [a]\neg\mathrm{On}(b) \rightarrow a = \mathrm{getOff}$$

So we get:

$$\Box\, [a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee (\mathrm{On}(b) \wedge \neg a = \mathrm{getOff})$$

Done! This is called a **successor-state axiom**.

Proof on paper

# Successor-State Axioms

**Definition: successor-state axiom**

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

$$\Box\, [a]F(\vec{x}) \leftrightarrow \gamma_F$$

or

$$\Box\, [a]f(\vec{x}) = y \leftrightarrow \gamma_f$$

where $\gamma_F, \gamma_f$ do not mention $\Box$ or $[t]$ operators.

# Successor-State Axioms

## Definition: successor-state axiom

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

$$\Box\,[a]F(\vec{x}) \leftrightarrow \gamma_F$$

or

$$\Box\,[a]f(\vec{x}) = y \leftrightarrow \gamma_f$$

where $\gamma_F, \gamma_f$ do not mention $\Box$ or $[t]$ operators.

Typical form of

- $\gamma_F$ is $\gamma_F^+ \vee (F(\vec{x}) \wedge \neg\gamma_F^-)$
- $\gamma_f$ is $\gamma_f^+ \vee (f(\vec{x}) = y \wedge \neg\exists y'\, {\gamma_f^+}^{\,y}_{\,y'})$

# Successor-State Axioms

## Definition: successor-state axiom

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

$$\Box\,[a]F(\vec{x}) \leftrightarrow \gamma_F$$

or

$$\Box\,[a]f(\vec{x}) = y \leftrightarrow \gamma_f$$

where $\gamma_F, \gamma_f$ do not mention $\Box$ or $[t]$ operators.

Typical form of

- $\gamma_F$ is $\gamma_F^+ \vee (F(\vec{x}) \wedge \neg\gamma_F^-)$
- $\gamma_f$ is $\gamma_f^+ \vee (f(\vec{x}) = y \wedge \neg\exists y'\, \gamma_f^+{}^{y}_{y'})$

Make sure that $\models \gamma_f{}^{y}_{y_1} \wedge \gamma_f{}^{y}_{y_2} \rightarrow y_1 = y_2$. Otherwise: inconsistency!

# Examples

- You're on a bus $\iff$ you got on it *or* you were on it and didn't get off it:

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

- Your position is $p$ $\iff$ you were on a bus that moved to $p$ *or* you were at $p$ already and not on a bus that moved:

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

# Basic Action Theories

An action theory must describe

- the initial situation

# Basic Action Theories

An action theory must describe

- the initial situation
- how fluents change $\Longrightarrow$ successor-state axioms

# Basic Action Theories

An action theory must describe

- the initial situation
- how fluents change $\Longrightarrow$ successor-state axioms
- the action preconditions $\Longrightarrow$ axiom for $\mathrm{Poss}(a)$

# Basic Action Theories

An action theory must describe

- the initial situation
- how fluents change $\implies$ successor-state axioms
- the action preconditions $\implies$ axiom for $\mathrm{Poss}(a)$

## Definition: basic action theory

$\Sigma_0 \wedge \Sigma_{\mathrm{dyn}}$ is a **basic action theory** over a set of fluents $\mathcal{F}$ iff

- $\Sigma_{\mathrm{dyn}}$ contains a successor-state axiom for every fluent in $\mathcal{F}$
- $\Sigma_{\mathrm{dyn}}$ contains an axiom $\Box\,\mathrm{Poss}(a) \leftrightarrow \pi$
- $\Sigma_0, \pi$ mention no $\mathrm{Poss}$, $\Box$, $[t]$.

# Example: the Bus Scenario as Basic Action Theory

$a$ = action, $b$ = bus, $d$ = destination, $p$ = position

- The initial situation:

  $\mathrm{pos} = \mathrm{Central} \wedge \mathrm{Route}(\mathrm{M50}, \mathrm{Uni})$

# Example: the Bus Scenario as Basic Action Theory

$a$ = action, $b$ = bus, $d$ = destination, $p$ = position

- The initial situation:

  $\text{pos} = \text{Central} \wedge \text{Route}(\text{M50}, \text{Uni})$

- You can get on/off a bus:

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

# Example: the Bus Scenario as Basic Action Theory

$a$ = action, $b$ = bus, $d$ = destination, $p$ = position

- The initial situation:

  $\mathrm{pos} = \mathrm{Central} \wedge \mathrm{Route}(\mathrm{M50}, \mathrm{Uni})$

- You can get on/off a bus:

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

- You can move by being on a bus that moves:

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

# Example: the Bus Scenario as Basic Action Theory

$a$ = action, $b$ = bus, $d$ = destination, $p$ = position

- The initial situation:

  $$\mathrm{pos} = \mathrm{Central} \wedge \mathrm{Route}(\mathrm{M50}, \mathrm{Uni})$$

- You can get on/off a bus:

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

- You can move by being on a bus that moves:

  $$\begin{aligned}\Box\, [a]\mathrm{pos} = p \leftrightarrow\ & \exists b \left(a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b)\right) \vee \\ & \left(\mathrm{pos} = p \wedge \neg \exists d\, \exists b\, (a = \mathrm{goTo}(b, d) \wedge \mathrm{On}(b))\right)\end{aligned}$$

- You can't get on (off) a bus when you're on one (none), and a bus can only go along its route:

  $$\begin{aligned}\Box\, \mathrm{Poss}(a) \leftrightarrow\ & \left(\exists b\, a = \mathrm{getOn}(b) \rightarrow \forall b\, \neg \mathrm{On}(b)\right) \wedge \\ & \left(a = \mathrm{getOff} \rightarrow \exists b\, \mathrm{On}(b)\right) \wedge \\ & \forall b \forall d \left(a = \mathrm{goTo}(b, d) \rightarrow \mathrm{Route}(b, d)\right)\end{aligned}$$

# The Projection Problem

The *central task* in reasoning about actions:

## Definition: projection problem

Given a basic action theory:
Is a goal formula true in a future situation?

$$\Sigma_0 \land \Sigma_{\mathrm{dyn}} \models [t_1] \ldots [t_j]\alpha$$

**Want:** a way to *eliminate* $[t]$ *operators*.

# The Projection Problem

The *central task* in reasoning about actions:

## Definition: projection problem

Given a basic action theory:
Is a goal formula true in a future situation?

$$\Sigma_0 \wedge \Sigma_{\text{dyn}} \models [t_1] \ldots [t_j]\alpha$$

**Want:** a way to *eliminate* $[t]$ *operators*.

Two approaches:

- Regression: reduce to $\Sigma_0 \models \alpha^*$
- Progression: reduce to $\Sigma_0^* \cup \Sigma_{\text{dyn}} \models \alpha$

# Overview of the Lecture

- Three Problems
- The Situation Calculus
- **Projection by regression**
- Projection by progression
- Knowledge and sensing
- Concluding words

# Regression – The Idea

- Successor state axioms relate truth after $a$ to truth before $a$:
  $\square\, [a]F(\vec{x}) \leftrightarrow \gamma_F$, where $\gamma_F$ mentions no $[t]$

# Regression – The Idea

- Successor state axioms relate truth after $a$ to truth before $a$:
  $\square\, [a]F(\vec{x}) \leftrightarrow \gamma_F$, where $\gamma_F$ mentions no $[t]$
- Idea: successively replace $[r]F(\vec{t})$ with $\gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}$

# Regression – The Idea

- Successor state axioms relate truth after $a$ to truth before $a$:
  $\Box\, [a]F(\vec{x}) \leftrightarrow \gamma_F$, where $\gamma_F$ mentions no $[t]$
- Idea: successively replace $[r]F(\vec{t})$ with $\gamma_F{}^{a\ \vec{x}}_{r\ \vec{t}}$
- Result: $\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models \alpha^*$

# Regression – The Idea

- Successor state axioms relate truth after $a$ to truth before $a$:
  $\Box\, [a]F(\vec{x}) \leftrightarrow \gamma_F$, where $\gamma_F$ mentions no $[t]$
- Idea: successively replace $[r]F(\vec{t})$ with $\gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}$
- Result: $\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models \alpha^*$
- Good: very simple and quite elegant

# Regression – The Idea

- Successor state axioms relate truth after $a$ to truth before $a$:
  $\square\,[a]F(\vec{x}) \leftrightarrow \gamma_F$, where $\gamma_F$ mentions no $[t]$
- Idea: successively replace $[r]F(\vec{t})$ with $\gamma_F{}^{a\ \vec{x}}_{r\ \vec{t}}$
- Result: $\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models \alpha^*$
- Good: very simple and quite elegant
- Bad: $\alpha^*$ may grow exponentially

# Regression

## Definition: regression operator, objective part

Regression of $\alpha$ is defined w.r.t. a basic action theory where $\gamma_F, \gamma_f$ are the RHSs of the successor-state axioms and $\pi$ is the RHS of the Poss axiom. We assume no variable in $\alpha$ is quantified twice in the same scope (as in $\exists x\,(\alpha \vee \exists x\,\beta)$):

- $\mathcal{R}[z \cdot r, F(\vec{t})] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_F{}^{a\ \vec{x}}_{r\ \vec{t}}]$
- $\mathcal{R}[z \cdot r, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_f{}^{a\ \vec{x}\ y}_{r\ \vec{t}\ t_0}]$ if $f$ is a function of sort object

# Regression

## Definition: regression operator, objective part

Regression of $\alpha$ is defined w.r.t. a basic action theory where $\gamma_F, \gamma_f$ are the RHSs of the successor-state axioms and $\pi$ is the RHS of the Poss axiom. We assume no variable in $\alpha$ is quantified twice in the same scope (as in $\exists x\,(\alpha \vee \exists x\,\beta)$):

- $\mathcal{R}[z \cdot r, F(\vec{t})] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_F{}^{a\;\vec{x}}_{r\;\vec{t}}]$
- $\mathcal{R}[z \cdot r, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_f{}^{a\;\vec{x}\;y}_{r\;\vec{t}\;t_0}]$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, F(\vec{t})] \stackrel{\text{def}}{=} F(\vec{t})$
- $\mathcal{R}[\langle\rangle, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} f(\vec{t}) = t_0$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, t_1 = t_0] \stackrel{\text{def}}{=} t_1 = t_0$ if $t_1$ is not a function of sort object

# Regression

## Definition: regression operator, objective part

Regression of $\alpha$ is defined w.r.t. a basic action theory where $\gamma_F, \gamma_f$ are the RHSs of the successor-state axioms and $\pi$ is the RHS of the Poss axiom. We assume no variable in $\alpha$ is quantified twice in the same scope (as in $\exists x\,(\alpha \vee \exists x\,\beta)$):

- $\mathcal{R}[z \cdot r, F(\vec{t})] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_F{}^{a\ \vec{x}}_{r\ \vec{t}}]$
- $\mathcal{R}[z \cdot r, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_f{}^{a\ \vec{x}\ y}_{r\ \vec{t}\ t_0}]$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, F(\vec{t})] \stackrel{\text{def}}{=} F(\vec{t})$
- $\mathcal{R}[\langle\rangle, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} f(\vec{t}) = t_0$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, t_1 = t_0] \stackrel{\text{def}}{=} t_1 = t_0$ if $t_1$ is not a function of sort object
- $\mathcal{R}[z, \text{Poss}(t)] \stackrel{\text{def}}{=} \mathcal{R}[z, \pi{}^{a}_{t}]$

# Regression

## Definition: regression operator, objective part

Regression of $\alpha$ is defined w.r.t. a basic action theory where $\gamma_F, \gamma_f$ are the RHSs of the successor-state axioms and $\pi$ is the RHS of the Poss axiom. We assume no variable in $\alpha$ is quantified twice in the same scope (as in $\exists x\,(\alpha \vee \exists x\,\beta)$):

- $\mathcal{R}[z \cdot r, F(\vec{t})] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}]$
- $\mathcal{R}[z \cdot r, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_f{}^{a\,\vec{x}\,y}_{r\,\vec{t}\,t_0}]$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, F(\vec{t})] \stackrel{\text{def}}{=} F(\vec{t})$
- $\mathcal{R}[\langle\rangle, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} f(\vec{t}) = t_0$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, t_1 = t_0] \stackrel{\text{def}}{=} t_1 = t_0$ if $t_1$ is not a function of sort object
- $\mathcal{R}[z, \mathrm{Poss}(t)] \stackrel{\text{def}}{=} \mathcal{R}[z, \pi{}^{a}_{t}]$
- $\mathcal{R}[z, (\alpha \vee \beta)] \stackrel{\text{def}}{=} (\mathcal{R}[z, \alpha] \vee \mathcal{R}[z, \beta])$
- $\mathcal{R}[z, \neg\alpha] \stackrel{\text{def}}{=} \neg\mathcal{R}[z, \alpha]$
- $\mathcal{R}[z, \exists x\,\alpha] \stackrel{\text{def}}{=} \exists x\,\mathcal{R}[z, \alpha]$

# Regression

## Definition: regression operator, objective part

Regression of $\alpha$ is defined w.r.t. a basic action theory where $\gamma_F, \gamma_f$ are the RHSs of the successor-state axioms and $\pi$ is the RHS of the $Poss$ axiom. We assume no variable in $\alpha$ is quantified twice in the same scope (as in $\exists x\,(\alpha \vee \exists x\,\beta)$):

- $\mathcal{R}[z \cdot r, F(\vec{t})] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}]$
- $\mathcal{R}[z \cdot r, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} \mathcal{R}[z, \gamma_f{}^{a\,\vec{x}\,y}_{r\,\vec{t}\,t_0}]$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, F(\vec{t})] \stackrel{\text{def}}{=} F(\vec{t})$
- $\mathcal{R}[\langle\rangle, f(\vec{t}) = t_0] \stackrel{\text{def}}{=} f(\vec{t}) = t_0$ if $f$ is a function of sort object
- $\mathcal{R}[\langle\rangle, t_1 = t_0] \stackrel{\text{def}}{=} t_1 = t_0$ if $t_1$ is not a function of sort object
- $\mathcal{R}[z, Poss(t)] \stackrel{\text{def}}{=} \mathcal{R}[z, \pi{}^{a}_{t}]$
- $\mathcal{R}[z, (\alpha \vee \beta)] \stackrel{\text{def}}{=} (\mathcal{R}[z, \alpha] \vee \mathcal{R}[z, \beta])$
- $\mathcal{R}[z, \neg\alpha] \stackrel{\text{def}}{=} \neg\mathcal{R}[z, \alpha]$
- $\mathcal{R}[z, \exists x\,\alpha] \stackrel{\text{def}}{=} \exists x\,\mathcal{R}[z, \alpha]$
- $\mathcal{R}[z, [t]\alpha] \stackrel{\text{def}}{=} \mathcal{R}[z \cdot t, \alpha]$

The first parameter in $\mathcal{R}[z, \alpha]$ is the “situation stack”.

# The Regression Result

**Theorem: regression**

Let $\Sigma_0 \wedge \Sigma_{\text{dyn}}$ be a basic action theory over $\mathcal{F}$.
Let $\alpha$ mention only fluents from $\mathcal{F} \cup \{\text{Poss}\}$ and no $\square$.

$$\Sigma_0 \cup \Sigma_{\text{dyn}} \models \alpha \iff \Sigma_0 \models \mathcal{R}[\langle\rangle, \alpha]$$

# Example

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

# Example

Let $\Sigma_0 \cup \Sigma_{\text{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\text{dyn}} \models [\text{getOn}(\text{M50})][\text{goTo}(\text{M50}, \text{Uni})]\text{pos} = \text{Uni}$ **?**

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\langle\rangle, [\text{getOn}(\text{M50})][\text{goTo}(\text{M50}, \text{Uni})]\text{pos} = \text{Uni}]$

# Example

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

$\Leftrightarrow \ \Sigma_0 \models \mathcal{R}[\langle\rangle, [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}]$

$\Leftrightarrow \ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}), \mathrm{pos} = \mathrm{Uni}]$

# Example

$$\Box\,[a]\mathrm{pos} = p \leftrightarrow \exists b\,\big(a = \mathrm{goTo}(b,p) \land \mathrm{On}(b)\big) \lor \big(\mathrm{pos} = p \land \neg\exists d\,\exists b\,(a = \mathrm{goTo}(b,d) \land \mathrm{On}(b))\big)$$

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\langle\rangle, [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}]$

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}), \mathrm{pos} = \mathrm{Uni}]$

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \gamma_{\mathrm{pos}}\,{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}}]$

# Example

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

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\langle\rangle, [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}]$

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}), \mathrm{pos} = \mathrm{Uni}]$

$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \gamma_{\mathrm{pos}}{}^{a}_{\mathrm{goTo(M50,Uni)}}{}^{p}_{\mathrm{Uni}}]$

$\Leftrightarrow\ \Sigma_0 \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \mathrm{On}(b)]\right) \vee \ \ldots$

# Example

$$\Box\, [a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \lor (\mathrm{On}(b) \land a \neq \mathrm{getOff})$$

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

$$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\langle\rangle, [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}]$$

$$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}), \mathrm{pos} = \mathrm{Uni}]$$

$$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \gamma_{\mathrm{pos}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}}]$$

$$\Leftrightarrow\ \Sigma_0 \models \exists b\, (\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \land \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \mathrm{On}(b)]) \lor \ldots$$

$$\Leftrightarrow\ \Sigma_0 \models \exists b\, (\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \land \mathcal{R}[\langle\rangle, \gamma_{\mathrm{On}}{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b}]) \lor \ldots$$

# Example

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

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

$$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\langle\rangle, [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}]$$

$$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}), \mathrm{pos} = \mathrm{Uni}]$$

$$\Leftrightarrow\ \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \gamma_{\mathrm{pos}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}}]$$

$$\Leftrightarrow\ \Sigma_0 \models \exists b\,\big(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \mathrm{On}(b)]\big) \vee \ldots$$

$$\Leftrightarrow\ \Sigma_0 \models \exists b\,\big(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}[\langle\rangle, \gamma_{\mathrm{On}}{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b}]\big) \vee \ldots$$

$$\begin{aligned}\Leftrightarrow\ \Sigma_0 \models \exists b\,\big(&\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \\ &\big(\mathrm{getOn}(\mathrm{M50}) = \mathrm{getOn}(b) \vee \\ &\ (\mathcal{R}[\langle\rangle, \mathrm{On}(b)] \wedge \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff})\big)\big) \vee \ldots\end{aligned}$$

# Example

Let $\Sigma_0 \cup \Sigma_{\mathrm{dyn}}$ be the bus scenario.

$\Sigma_0 \cup \Sigma_{\mathrm{dyn}} \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}$ **?**

$$\Leftrightarrow \Sigma_0 \models \mathcal{R}[\langle\rangle, [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{pos} = \mathrm{Uni}]$$

$$\Leftrightarrow \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}) \cdot \mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}), \mathrm{pos} = \mathrm{Uni}]$$

$$\Leftrightarrow \Sigma_0 \models \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \gamma_{\mathrm{pos}}\,{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}}]$$

$$\Leftrightarrow \Sigma_0 \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}[\mathrm{getOn}(\mathrm{M50}), \mathrm{On}(b)]\right) \vee \ldots$$

$$\Leftrightarrow \Sigma_0 \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}[\langle\rangle, \gamma_{\mathrm{On}}\,{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b}]\right) \vee \ldots$$

$$\Leftrightarrow \Sigma_0 \models \exists b \big(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \\ \big(\mathrm{getOn}(\mathrm{M50}) = \mathrm{getOn}(b) \vee \\ (\mathcal{R}[\langle\rangle, \mathrm{On}(b)] \wedge \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff})\big)\big) \vee \ldots$$

$$\Leftrightarrow \Sigma_0 \models \exists b \underbrace{\left(\mathrm{M50} = b \wedge \left(\mathrm{M50} = b \vee \mathcal{R}[\langle\rangle, \mathrm{On}(b)]\right)\right)}_{\text{Valid if } b \text{ is M50. So the whole formula is valid and hence entailed by } \Sigma_0.} \vee \ldots \quad ✓$$

# Overview of the Lecture

- Three Problems
- The Situation Calculus
- Projection by regression
- **Projection by progression**
- Knowledge and sensing
- Concluding words

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$
- Idea: use $\gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}$ to initialise new $F(\vec{t})$, forget old $F(\vec{t})$

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$
- Idea: use $\gamma_F{}^{a}_{r}{}^{\vec{x}}_{\vec{t}}$ to initialise new $F(\vec{t})$, forget old $F(\vec{t})$
- Result: $\Sigma_0 \cup \Sigma_{\text{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0^* \cup \Sigma_{\text{dyn}} \models \alpha$

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$
- Idea: use $\gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}$ to initialise new $F(\vec{t})$, forget old $F(\vec{t})$
- Result: $\Sigma_0 \cup \Sigma_{\text{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0^* \cup \Sigma_{\text{dyn}} \models \alpha$
- Progression is the *dual* to regression

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$
- Idea: use $\gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}$ to initialise new $F(\vec{t})$, forget old $F(\vec{t})$
- Result: $\Sigma_0 \cup \Sigma_{\text{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0^* \cup \Sigma_{\text{dyn}} \models \alpha$
- Progression is the *dual* to regression
- Big problem: forgetting is very hard to formalise!
  - Requires second-order logic in general
  - Second-order logic features quantification over predicates/functions

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$
- Idea: use $\gamma_F{}^{a\,\vec{x}}_{r\,\vec{t}}$ to initialise new $F(\vec{t})$, forget old $F(\vec{t})$
- Result: $\Sigma_0 \cup \Sigma_{\text{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0^* \cup \Sigma_{\text{dyn}} \models \alpha$
- Progression is the *dual* to regression
- Big problem: forgetting is very hard to formalise!
  - Requires second-order logic in general
  - Second-order logic features quantification over predicates/functions
  - Actions like $\text{goTo}(b, d)$ cause the problem
    - $\text{goTo}(b, d)$ moves the passengers of the bus
    - Indirect effects

# Progression – The Idea

- Want a new $\Sigma_0$ *after* action $t$
- Idea: use $\gamma_F {}^{a\,\vec{x}}_{r\,\vec{t}}$ to initialise new $F(\vec{t})$, forget old $F(\vec{t})$
- Result: $\Sigma_0 \cup \Sigma_{\text{dyn}} \models [t_1] \ldots [t_j]\alpha$ reduces to $\Sigma_0^* \cup \Sigma_{\text{dyn}} \models \alpha$
- Progression is the *dual* to regression
- Big problem: forgetting is very hard to formalise!
  - Requires second-order logic in general
  - Second-order logic features quantification over predicates/functions
  - Actions like $\text{goTo}(b, d)$ cause the problem
    - $\text{goTo}(b, d)$ moves the passengers of the bus
    - Indirect effects
  - Expressible subclasses are known

# Overview of the Lecture

- Three Problems
- The Situation Calculus
- Projection by regression
- Projection by progression
- **Knowledge and sensing**
- Concluding words

# Knowledge and Sensing

- New formulas: $\mathbf{K}\alpha \quad \mathbf{O}\alpha$
- Predicate $\mathrm{SF}(t)$ represents sensing result of action $t$

# Knowledge and Sensing

- New formulas: $\mathbf{K}\alpha \quad \mathbf{O}\alpha$
- Predicate $\mathrm{SF}(t)$ represents sensing result of action $t$

Ex.: You ask the driver whether the bus is going to UNSW

- "Yes" $\Longrightarrow$ you know the bus going to UNSW
- "No" $\Longrightarrow$ you know the bus is not going to UNSW

# Knowledge and Sensing

- New formulas: $\mathbf{K}\alpha \quad \mathbf{O}\alpha$
- Predicate $\mathrm{SF}(t)$ represents sensing result of action $t$

Ex.: You ask the driver whether the bus is going to UNSW

- "Yes" $\Longrightarrow$ you know the bus going to UNSW
- "No" $\Longrightarrow$ you know the bus is not going to UNSW

Formalisation of knowledge and sensing:

- Set of possible worlds $e$
- Doing $A$ tells you the value of $\mathrm{SF}(A)$ in real world $w$
- Only consider those $w' \in e$ which agree with $w$

  If $w$ says bus goes to UNSW, only consider $w'$ where bus goes to UNSW

# The Semantics of Knowledge and Sensing

## Definition: semantics of knowledge and sensing

$w \simeq_z w' \iff w, w'$ agree on the sensing results:

- $w \simeq_{\langle\rangle} w'$
- $w \simeq_{z \cdot n} w' \iff w \simeq_z w'$ and $w[\mathrm{SF}(n), z] = w'[\mathrm{SF}(n), z]$

# The Semantics of Knowledge and Sensing

## Definition: semantics of knowledge and sensing

$w \simeq_z w' \iff w, w'$ agree on the sensing results:

- $w \simeq_{\langle\rangle} w'$
- $w \simeq_{z \cdot n} w' \iff w \simeq_z w'$ and $w[\mathrm{SF}(n), z] = w'[\mathrm{SF}(n), z]$

An **epistemic state** $e$ is a set of worlds.

# The Semantics of Knowledge and Sensing

**Definition: semantics of knowledge and sensing**

$w \simeq_z w' \iff w, w'$ agree on the sensing results:

- $w \simeq_{\langle\rangle} w'$
- $w \simeq_{z \cdot n} w' \iff w \simeq_z w'$ and $w[\mathrm{SF}(n), z] = w'[\mathrm{SF}(n), z]$

An **epistemic state** $e$ is a set of worlds.

- Rules from Slide 14 retrofitted with additional $e$ parameter, e.g., $e, w, z \models \neg\alpha \iff e, w, z \not\models \alpha$

# The Semantics of Knowledge and Sensing

## Definition: semantics of knowledge and sensing

$w \simeq_z w' \iff w, w'$ agree on the sensing results:

- $w \simeq_{\langle\rangle} w'$
- $w \simeq_{z \cdot n} w' \iff w \simeq_z w'$ and $w[\mathrm{SF}(n), z] = w'[\mathrm{SF}(n), z]$

An **epistemic state** $e$ is a set of worlds.

- Rules from Slide 14 retrofitted with additional $e$ parameter, e.g., $e, w, z \models \neg\alpha \iff e, w, z \not\models \alpha$
- $e, w, z \models \mathbf{K}\alpha \iff$ for all worlds $w'$,
  $w' \in e$ and $w \simeq_z w' \Rightarrow e, w', z \models \alpha$

# The Semantics of Knowledge and Sensing

## Definition: semantics of knowledge and sensing

$w \simeq_z w' \iff w, w'$ agree on the sensing results:

- $w \simeq_{\langle\rangle} w'$
- $w \simeq_{z \cdot n} w' \iff w \simeq_z w'$ and $w[\mathrm{SF}(n), z] = w'[\mathrm{SF}(n), z]$

An **epistemic state** $e$ is a set of worlds.

- Rules from Slide 14 retrofitted with additional $e$ parameter, e.g., $e, w, z \models \neg\alpha \iff e, w, z \not\models \alpha$
- $e, w, z \models \mathbf{K}\alpha \iff$ for all worlds $w'$,
  $w' \in e$ and $w \simeq_z w' \Rightarrow e, w', z \models \alpha$
- $e, w, z \models \mathbf{O}\alpha \iff$ for all worlds $w'$,
  $w' \in e$ and $w \simeq_z w' \Leftrightarrow e, w', z \models \alpha$

$\Sigma \models \alpha \iff$ for all $e, w$, if $e, w, \langle\rangle \models \beta$ for all $\beta \in \Sigma$, then $e, w, \langle\rangle \models \alpha$

# Basic Action Theories with Knowledge

An action theory must describe

- what is true the initial situation
- what is *known* about the initial situation
- how fluents change $\implies$ successor-state axioms
- the action preconditions $\implies$ axiom for $\mathrm{Poss}(a)$
- how *sensing* works $\implies$ axiom for $\mathrm{SF}(a)$

# Basic Action Theories with Knowledge

An action theory must describe

- what is true the initial situation
- what is *known* about the initial situation
- how fluents change $\Longrightarrow$ successor-state axioms
- the action preconditions $\Longrightarrow$ axiom for $\mathrm{Poss}(a)$
- how *sensing* works $\Longrightarrow$ axiom for $\mathrm{SF}(a)$

## Definition: basic action theory

$\Sigma_0 \wedge \Sigma_{\mathrm{dyn}} \wedge \mathbf{O}(\Sigma_1 \wedge \Sigma_{\mathrm{dyn}})$ is a **basic action theory** over $\mathcal{F}$ iff

- $\Sigma_{\mathrm{dyn}}$ contains a successor-state axiom for every fluent in $\mathcal{F}$
- $\Sigma_{\mathrm{dyn}}$ contains an axiom $\Box\,\mathrm{Poss}(a) \leftrightarrow \pi$
- $\Sigma_{\mathrm{dyn}}$ contains an axiom $\Box\,\mathrm{SF}(a) \leftrightarrow \varphi$
- $\Sigma_0, \Sigma_1, \pi, \varphi$ mention no $\mathrm{Poss}$, $\mathrm{SF}$, $\Box$, $[t]$.

# Example: the Bus Scenario as Basic Action Theory

- What is true, what is known initially:

$$\Sigma_0 \stackrel{\text{def}}{=} \text{pos} = \text{Central} \land \text{Route}(\text{M50}, \text{Uni})$$

$$\Sigma_1 \stackrel{\text{def}}{=} \text{pos} = \text{Central}$$

# Example: the Bus Scenario as Basic Action Theory

- What is true, what is known initially:

  $\Sigma_0 \stackrel{\text{def}}{=} \text{pos} = \text{Central} \wedge \text{Route}(\text{M50}, \text{Uni})$

  $\Sigma_1 \stackrel{\text{def}}{=} \text{pos} = \text{Central}$

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

- $$\Box\, [a]\text{pos} = p \leftrightarrow \exists b \left(a = \text{goTo}(b, p) \wedge \text{On}(b)\right) \vee \left(\text{pos} = p \wedge \neg\exists d\, \exists b\, (a = \text{goTo}(b, d) \wedge \text{On}(b))\right)$$

- $$\Box\, \text{Poss}(a) \leftrightarrow \left(\exists b\, a = \text{getOn}(b) \rightarrow \forall b\, \neg\text{On}(b)\right) \wedge \left(a = \text{getOff} \rightarrow \exists b\, \text{On}(b)\right) \wedge \forall b \forall d \left(a = \text{goTo}(b, d) \rightarrow \text{Route}(b, d)\right)$$

# Example: the Bus Scenario as Basic Action Theory

- What is true, what is known initially:

  $\Sigma_0 \stackrel{\text{def}}{=} \text{pos} = \text{Central} \wedge \text{Route}(\text{M50}, \text{Uni})$

  $\Sigma_1 \stackrel{\text{def}}{=} \text{pos} = \text{Central}$

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

- $$\Box\, [a]\text{pos} = p \leftrightarrow \exists b \left(a = \text{goTo}(b, p) \wedge \text{On}(b)\right) \vee \left(\text{pos} = p \wedge \neg\exists d\, \exists b\, (a = \text{goTo}(b, d) \wedge \text{On}(b))\right)$$

- $$\Box\, \text{Poss}(a) \leftrightarrow \left(\exists b\, a = \text{getOn}(b) \rightarrow \forall b\, \neg\text{On}(b)\right) \wedge \left(a = \text{getOff} \rightarrow \exists b\, \text{On}(b)\right) \wedge \forall b \forall d \left(a = \text{goTo}(b, d) \rightarrow \text{Route}(b, d)\right)$$

- You can ask and learn whether the bus stops at a destination:

  $\Box\, \text{SF}(a) \leftrightarrow \forall b \forall d \left(a = \text{ask}(b, d) \rightarrow \text{Route}(b, d)\right)$

# Regression of Knowledge

**Theorem: knowledge after action**

$$\models [a]\mathbf{K}\alpha \leftrightarrow (\mathrm{SF}(a) \rightarrow \mathbf{K}(\mathrm{SF}(a) \rightarrow [a]\alpha)) \wedge$$
$$(\neg\mathrm{SF}(a) \rightarrow \mathbf{K}(\neg\mathrm{SF}(a) \rightarrow [a]\alpha))$$

Looks like a successor-state axiom, but it's a *theorem*!

# Regression of Knowledge

**Theorem: knowledge after action**

$$\models [a]\mathbf{K}\alpha \leftrightarrow (\mathrm{SF}(a) \rightarrow \mathbf{K}(\mathrm{SF}(a) \rightarrow [a]\alpha)) \wedge$$
$$(\neg\mathrm{SF}(a) \rightarrow \mathbf{K}(\neg\mathrm{SF}(a) \rightarrow [a]\alpha))$$

Looks like a successor-state axiom, but it's a *theorem*!

**Definition: regression operator, subjective part**

- $\mathcal{R}[\langle\rangle, \mathbf{K}\alpha] \stackrel{\text{def}}{=} \mathbf{K}\mathcal{R}[\langle\rangle, \alpha]$
- $\mathcal{R}[z \cdot r, \mathbf{K}\alpha] \stackrel{\text{def}}{=} \mathcal{R}[z, \big(\mathrm{SF}(r) \rightarrow \mathbf{K}(\mathrm{SF}(r) \rightarrow [r]\alpha)\big)] \wedge$
  $\mathcal{R}[z, \big(\neg\mathrm{SF}(r) \rightarrow \mathbf{K}(\neg\mathrm{SF}(r) \rightarrow [r]\alpha)\big)]$
- $\mathcal{R}[z, \mathrm{SF}(t)] \stackrel{\text{def}}{=} \mathcal{R}[z, \varphi_t^a]$

# The Regression Result with Knowledge

**Theorem: regression**

Let $\Sigma_0 \land \Sigma_{\mathrm{dyn}} \land \mathbf{O}(\Sigma_1 \land \Sigma_{\mathrm{dyn}})$ be a basic action theory over $\mathcal{F}$.
Let $\alpha$ mention only fluents from $\mathcal{F} \cup \{\mathrm{Poss}, \mathrm{SF}\}$ and no $\mathbf{O}$ or $\Box$.

$$\Sigma_0 \land \Sigma_{\mathrm{dyn}} \land \mathbf{O}(\Sigma_1 \land \Sigma_{\mathrm{dyn}}) \models \alpha \iff \Sigma_0 \land \mathbf{O}\Sigma_1 \models \mathcal{R}[\langle\rangle, \alpha]$$

# The Regression Result with Knowledge

**Theorem: regression**

Let $\Sigma_0 \wedge \Sigma_{\text{dyn}} \wedge \mathbf{O}(\Sigma_1 \wedge \Sigma_{\text{dyn}})$ be a basic action theory over $\mathcal{F}$.
Let $\alpha$ mention only fluents from $\mathcal{F} \cup \{\text{Poss}, \text{SF}\}$ and no $\mathbf{O}$ or $\Box$.

$$\Sigma_0 \wedge \Sigma_{\text{dyn}} \wedge \mathbf{O}(\Sigma_1 \wedge \Sigma_{\text{dyn}}) \models \alpha \iff \Sigma_0 \wedge \mathbf{O}\Sigma_1 \models \mathcal{R}[\langle\rangle, \alpha]$$

Reasoning about actions + knowledge

+ Regression (eliminates $[t]$)
+ Representation theorem (eliminates $\mathbf{K}$)

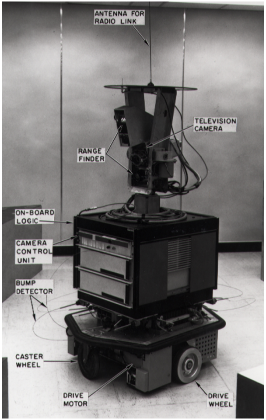
= Non-modal reasoning!

# Overview of the Lecture

- Three Problems
- The Situation Calculus
- Projection by regression
- Projection by progression
- Knowledge and sensing
- **Concluding words**

# Relationship to Planning

- Modelling dynamic systems is core AI
- In the beginning (1950ies, 1960ies):
  reasoning about action = planning
- McCarthy's situation calculus (1963, 1969):
  too expressive, impractical
- Shakey introduced STRIPS for planning
- Reasoning about action and planning diverged
- Past years: they converge again
  - Reasoning action gets more efficient
  - Planning gets more expressive
  - Both sides benefit



# Relevant Questions?

**Reasoning about Knowledge**

- Why not classical logic?

**Semantics of knowledge**

- How is $\mathbf{K}\alpha$ defined?
- How is $\mathbf{O}\alpha$ defined?
- How does quantification work?

**Knowing that vs knowing what/who**

- What's the difference?
- Why is that semantic difference?

**Representation theorem**

- What are known instances?
- How does RES do it?

**Logical Omniscience**

- What does it mean?
- Why is it a problem?

**Limited belief I**

- Why more worlds?
- What is true/false support?
- When good/bad complexity?
- Why?

**Limited belief II**

- What's unit propagation?
- What's subsumption?
- How is $\mathbf{K}_k\alpha$ defined?
- Soundness vs completeness?

**Implementation**

- How does DPLL work?
- Idea behind watched lits?
- Idea behind CDCL?

**Reasoning about actions**

- What are the problems?

**Solution of frame problem**

- What's a succ.-state axiom?
- What's a basic action theory?

**Projection**

- What's the projection task?
- What are the approaches?
- How does regression work?

**Semantics of actions**

- How are worlds defined?
- What does $\mathrm{SF}(t)$ mean?
- How is $\mathbf{K}\alpha$ defined in sitcalc?

This list is not intended to be exhaustive.