# KNOWLEDGE REPRESENTATION, REASONING AND DECLARATIVE PROBLEM SOLVING

By Jicheng Zhao
based on the notes of others

Class 16: Action Language in Explanation and Planning

# Temporal projection with incomplete initial state

- Recall our usual example of code written in the action language $A$.
  - Domain:

$$a\ \ auses\ f\ if\ p_1, ..., p_m, \neg q_1, ..., \neg q_n.$$
$$a\ \ causes\ \neg g\ if\ p_1, ..., p_m, \neg q_1, ..., \neg q_n.$$

  - Observation:

$$initially\ f.$$
$$initially\ \neg g.$$

  - Query: $f\ after\ a_1, ..., a_m.$
- We translated these rules into AnsProlog$^{\neg}$:

– Domain:

$$\begin{aligned}
holds(f, res(a, s)) &\leftarrow holds(p_1, s)...holds(p_m, s), \\
&\quad \neg holds(q_1, s)...\neg holds(q_n, s). \\
ab(f, a, s) &\leftarrow not\ \neg holds(p, s). \\
\neg holds(f, res(a, s)) &\leftarrow holds(p_1, s)...holds(p_m, s), \\
&\quad \neg holds(q_1, s)...\neg holds(q_n, s). \\
ab(f, a, s) &\leftarrow not\ holds(p, s).
\end{aligned}$$

– Observation:

$$\begin{aligned}
holds(f, s_0). \\
\neg holds(f, s_0).
\end{aligned}$$

– The Rules for Inertia:

$$\begin{aligned}
holds(F, res(A, S)) &\leftarrow holds(F, S),\ not\ ab(F, A, S). \\
\neg holds(F, res(A, S)) &\leftarrow holds(F, S),\ not\ ab(F, A, S).
\end{aligned}$$

- If the initial state is incomplete, we need to define splitting rules

$$holds(F, s_0) \leftarrow not\ \neg holds(F, s_0).$$

$$\neg holds(F, s_0) \leftarrow not\ holds(F, s_0).$$

$\forall\ fluents\ F$

By this construct, we gain some expression from our previous version, which wasn't *sound.* With this formulation, answer set results are similar to:

$$a \leftarrow not\ b \qquad (1)$$

$$b \leftarrow not\ a \qquad (2)$$

which yields two answer sets.

- This is important if we need to make certain inferences about:

$$a\ causes\ f\ if\ p$$

$$a\ causes\ f\ if\ \neg p$$

$f$ should be true after exectuing action $a$ by intuition but the old formulation will not be able to infer that as we discussed in the previous lecture whereas the method with splitting rules will.

# Explanations

- Here we are given an observation, and we want a set of actions that can explain the observation. We can use our previous formulation by adding constraints.

  **Example 1** *We observe* $\neg alive\ after$ *shoot and should thus throw out any answer sets with the term* $holds(alive, res(shoot, s_0))$ *by using the constraint:*

  $$\leftarrow \neg holds(alive, res(shoot, s_0)).$$

- Hence, to handle observations, we enumerate all possibilities and formulate the observations as constraints to eliminate answer sets that disagree. Then, look at one explanation at a time.

# Planning

- Find $x_1 ... x_k$ such that $(D, O)$ entails $goal\ after x_1 ... x_k$.
- $O$ is complete - for every fluent we know exactly if that fluent is true or false in the initial state.
- Initially, we though that we could do it with a "satisfied" rule:

$$\begin{aligned} satisfied &\leftarrow holds(goal, res(x_k ... res(x_1, s_0) ...)) \\ &\leftarrow \textbf{not}\ satisfied \end{aligned}$$

- This approach works. However, the approach is not feasible due to the complexity of the atoms.

- We will change our formulation to chain together events.

$$\begin{aligned} holds(f, T+1) \;\leftarrow\; & occurs(a, T), holds(p_1, T) ... holds(p_m, T), \\ & \neg holds(q_1, T) ... \neg holds(q_n, T). \end{aligned}$$

- We can then use cardinality constraints (or normal constraints) to ensure that exactly one action occurs at each time.

$$otherOccurs(A, T) \leftarrow occurs(B, T), B \neq A\ , time(T).$$
$$occurs(A, T) \leftarrow not\ otherOccurs(A, T), time(T).$$

Alternatively, we can encode with weighted logic program:

$$1\,\{occurs(A, T) : action(A)\}\,1 : -time(T).$$

- Why does this formulation only work for complete initial state?