

Presented by Yilan Gu
AAAI-08, July 17th, 2008

Reasoning about Large Taxonomies of Actions

Yilan Gu

Dept. of Computer Science
University of Toronto
Toronto, ON, Canada

Mikhail Soutchanski

Dept. of Computer Science
Ryerson University
Toronto, ON, Canada

Outline

- Background: the Situation Calculus
- Motivation
- Action Hierarchies
- Constructing a Taxonomy of Actions
- Modular Basic Action Theories
- Correctness and Computational Advantages
- Conclusion and Future work

The Situation Calculus (Basic Concepts)

- Three sorts

- Action function: $\text{cook}(x)$, $\text{deepFry}(x,y)$, $\text{heatFood}(x)$, ...
- Situation: S_0 , $\text{do}(a,s)$,
 $\text{do}([a_1, \dots, a_n],s) = \text{do}(a_n, \dots, \text{do}(a_1,s))$
- Object: potato , fish , ...

- Fluents

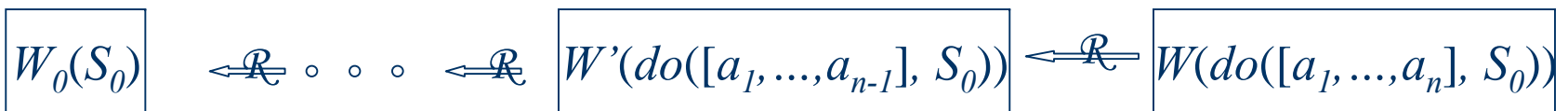
- system features whose truth values may vary
- Examples: $\text{cooked}(x,s)$, $\text{warm}(x,s)$, $\text{boiled}(x,s)$, $\text{fried}(x,s)$, ...

A Basic Action Theory (BAT) \mathcal{D}

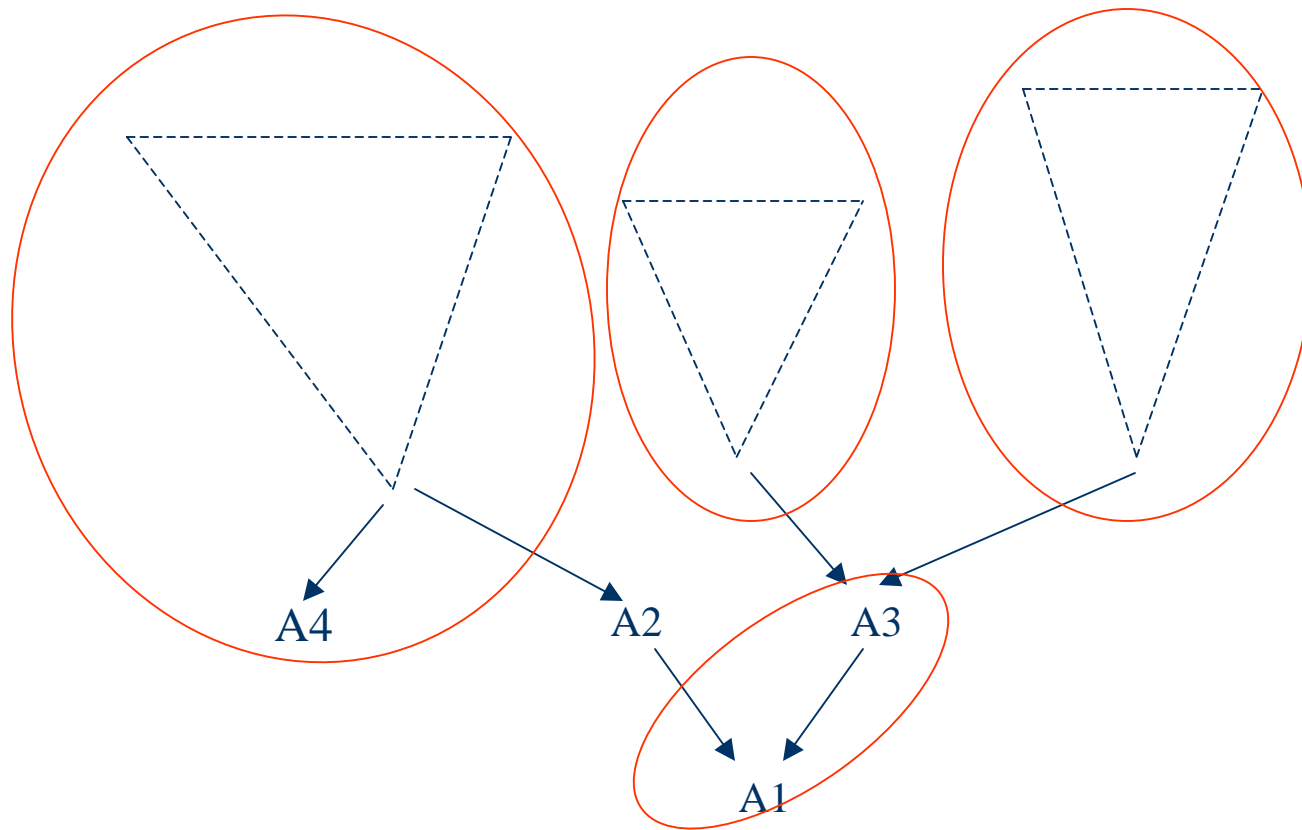
- Precondition axioms for actions \mathcal{D}_{ap} -- One axiom per action function
E.g., $\text{Poss}(\text{fry}(x,y),s) \equiv \text{friableObject}(x) \wedge \text{fryingVessel}(y) \wedge \text{contain}(y,\text{Oil},s)$.
- Successor state axioms \mathcal{D}_{ss} -- One axiom per fluent
E.g., $\text{contain}(x,y,\text{do}(a,s)) \equiv a=\text{add}(y,x) \vee \text{contain}(x,y,s) \wedge \neg(a=\text{remove}(y,x))$.
- Axioms for initial database \mathcal{D}_{S_0}
E.g., $\text{friableObject}(\text{Egg}). \text{fryingVessel}(\text{FryingPan}). \neg\text{contain}(\text{Oil},\text{FryingPan},S_0)$.
- Unique name axioms for actions \mathcal{D}_{una}
E.g., $\text{cook}(x) \neq \text{heatFood}(y). \quad \text{fry}(x,y) \neq \text{cook}(z).$
 $\text{fry}(x,y) = \text{fry}(z,o) \supset x=z \wedge y=o.$
- Foundational axioms for situations Σ -- properties for situations
E.g., $S_0 \neq \text{do}(a,s). \quad \text{do}(a,s) = \text{do}(a',s') \supset a=a' \wedge s=s'.$

Reasoning about actions -- regression

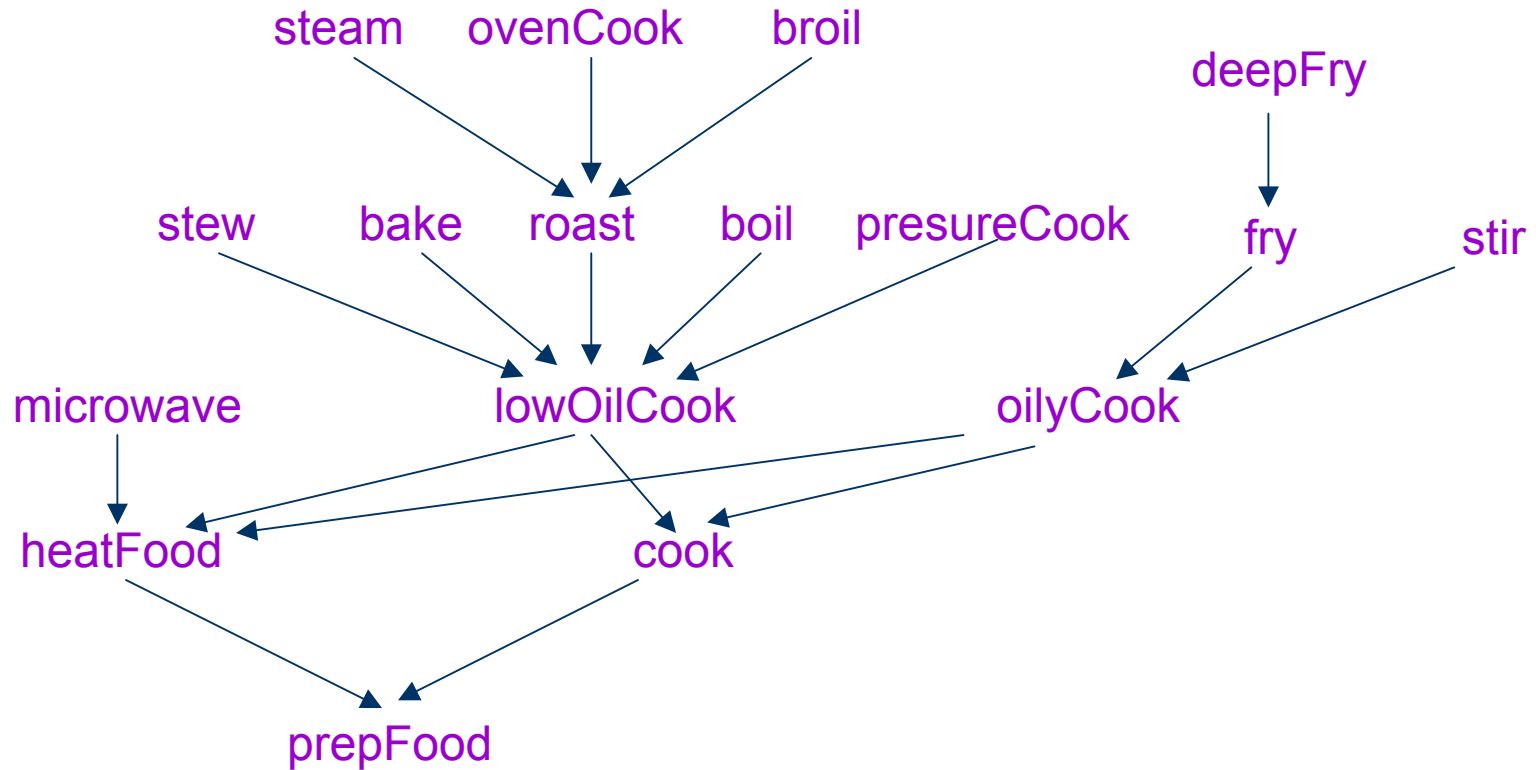
- Key reasoning mechanism -- regression operator \mathcal{R}
 - Successor state axioms support regression in a natural way:
If $F(x, do(a, s)) \equiv \Psi_F(x, a, s)$, then $\mathcal{R}[F(t, do(A, S))] = \mathcal{R}[\Psi_F(t, A, S)]$.
 - Regression defined recursively, in particular:
E.g., $\mathcal{R}[W_1 \vee W_2] = \mathcal{R}[W_1] \vee \mathcal{R}[W_2]$
- Projection problem:
given an goal W and a sequence of actions A_1, \dots, A_n , whether
 $\mathcal{D} \models W(do([A_n, \dots, A_1], S_0))$
- Regression Theorem: $\mathcal{D} \models W$ iff $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W]$



Motivation

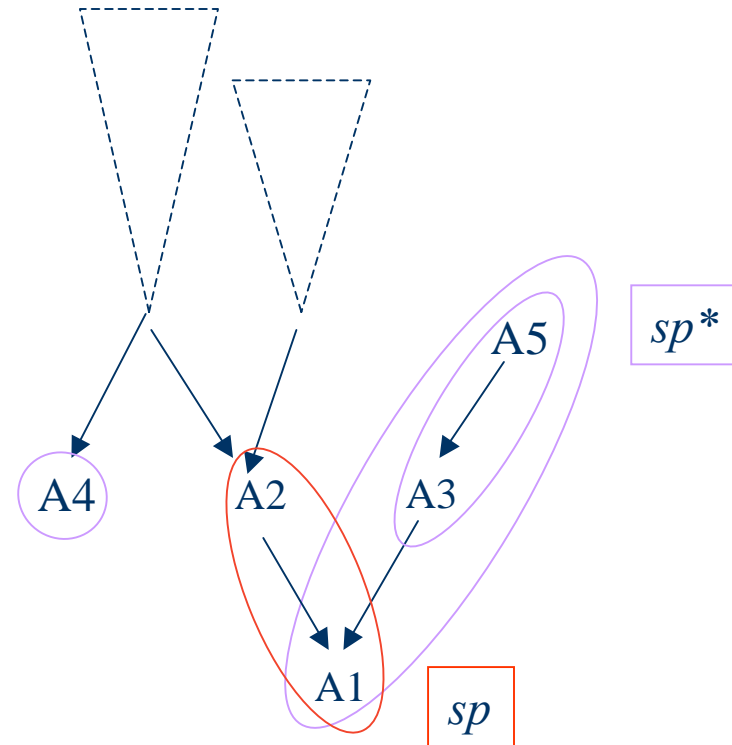


A Cooking Example



Action Hierarchy \mathcal{H}^*

- Finitely many axioms:
 $sp(A_2(\vec{x}), A_1(\vec{y})) \equiv \Phi(\vec{x}, \vec{y})$,
where Φ is not unsatisfiable
- sp^* -- reflexive-transitive closure of sp defined in second-order logic
- Acyclic directed graph:
 $A_2 \rightarrow A_1$ iff there is an axiom
 $sp(A_2(\vec{x}), A_1(\vec{y})) \equiv \Phi(\vec{x}, \vec{y})$
- Multiple inheritance



Examples of Action Hierarchy Axioms

- Examples of direct specializations:

$sp(\text{heatFood}(x), \text{prepFood}(x)).$ $sp(\text{cook}(x), \text{prepFood}(x)).$
 $sp(\text{microwave}(x), \text{heatFood}(x)).$ $sp(\text{lowOilcook}(x), \text{heatFood}(x)).$
 $sp(\text{oilyCook}(x), \text{cook}(x)).$ $sp(\text{stew}(x,y), \text{lowOilCook}(x)).$
... ..
 $sp(\text{deepFry}(x,y), \text{fry}(x,y)).$

- Examples of (distant) specializations:

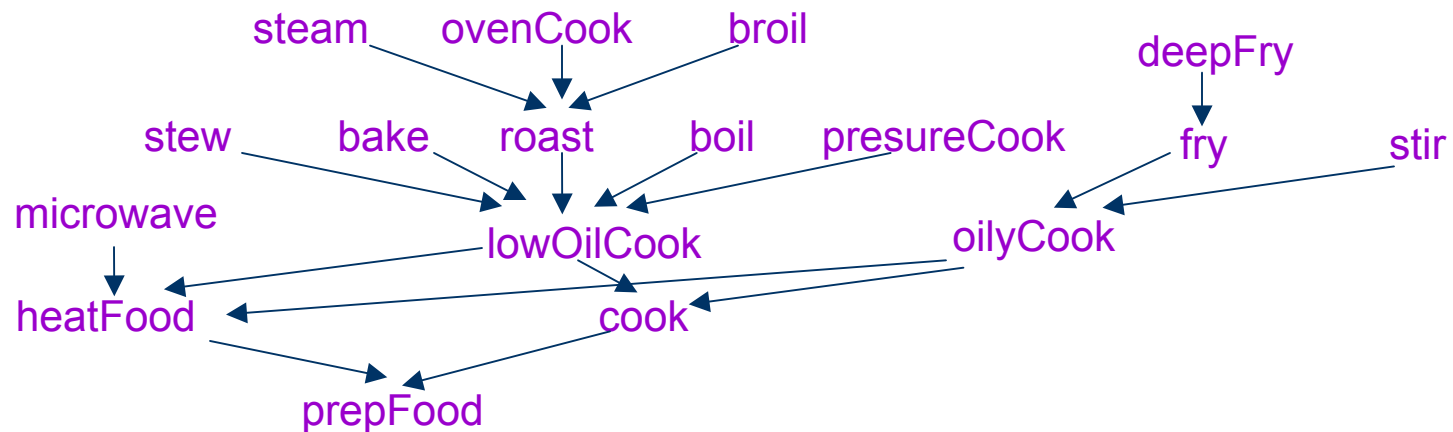
$sp^*(\text{cook}(x), \text{cook}(x)).$ $sp^*(\text{deepFry}(x,y), \text{fry}(x,y)).$
 $sp^*(\text{deepFry}(x,y), \text{prepFood}(x)).$
... ..

How to Construct a Taxonomy of Actions

- Action hierarchies with monotonic inheritance of effects, i.e.,
 $\mathcal{D}^{\mathcal{H}} \models (\forall F). \text{sp}^*(a_1, a) \wedge \neg (F(s) \equiv F(\text{do}(a, s))) \supset F(\text{do}(a_1, s)) \equiv F(\text{do}(a, s)).$
- Given a set of actions and finitely many fluents, the effects of actions on fluents are known
- General idea of constructing a taxonomy of actions
 - Sort action functions in an order $A_1(\vec{x}_1), \dots, A_n(\vec{x}_n)$ such that $A_1(\vec{x}_1)$ affects the least number of fluents and $A_n(\vec{x}_n)$ affects the most number of fluents
 - Assume that the direct specialization relations among $A_1(\vec{x}_1), \dots, A_k(\vec{x}_k)$ ($1 \leq k \leq n-1$), have been developed, starting from $A_k(\vec{x}_k)$, look for conditions under which $A_{k+1}(\vec{x}_{k+1})$ causes same effects on all fluents that can be affected by $A_i(\vec{x}_i)$ ($1 \leq i \leq k$)
 - Continue until the end of the sequence

An Example of Constructing a Taxonomy of Actions

Sorting: $\text{prepFood}(x_1), \text{heatFood}(x_2), \text{cook}(x_3), \dots, \text{broil}(x_{16}), \text{deepFry}(x_{17})$



$\text{sp}(\text{heatFood}(x_2), \text{prepFood}(x_1)) \equiv x_1=x_2$ (i.e., $\text{sp}(\text{heatFood}(x), \text{prepFood}(x))$.)

$\text{sp}(\text{cook}(x), \text{prepFood}(x))$.

$\text{sp}(\text{microwave}(x), \text{heatFood}(x))$.

$\text{sp}(\text{lowOilCook}(x), \text{cook}(x))$.

$\text{sp}(\text{lowOilCook}(x), \text{heatFood}(x))$.

... ..

$\text{sp}(\text{broil}(x,y), \text{roast}(x,y))$.

$\text{sp}(\text{deepFry}(x,y), \text{fry}(x,y))$.

Modular BAT Representation

- A modular BAT $\mathcal{D}^{\mathcal{H}} = \mathcal{D}_{S0}^{\mathcal{H}} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss}^{\mathcal{H}} \cup \mathcal{D}_{una} \cup \Sigma$
 1. $\mathcal{D}_{S0}^{\mathcal{H}}$ – the usual initial theory plus the specialization axioms and the definition of sp^* , i.e., $\mathcal{D}_{S0}^{\mathcal{H}} = \mathcal{D}_{S0} \cup \mathcal{H}^*$
 2. $\mathcal{D}_{ss}^{\mathcal{H}}$ – the modular successor state axioms
Positive and negative effects can be specified for groups of actions using sp^*
 3. \mathcal{D}_{ap} , \mathcal{D}_{una} , Σ – same as Reiter's axioms

Comparing with Reiter's SSAs (Example)

- Our representation of SSAs

$\text{cooked}(x, \text{do}(a, s)) \equiv \text{sp}^*(a, \text{cook}(x)) \vee \text{cooked}(x, s).$

- Reiter's representation of SSAs

$\text{cooked}(x, \text{do}(a, s)) \equiv$

$a = \text{cook}(x) \vee a = \text{lowOilCook}(x) \vee a = \text{oilyCook}(x) \vee$
 $a = \text{steam}(x, y) \vee a = \text{boil}(x, y) \vee a = \text{stew}(x, y) \vee$
 $a = \text{broil}(x, y) \vee a = \text{bake}(x, y) \vee a = \text{ovenCook}(x, y) \vee$
 $a = \text{roast}(x, y) \vee a = \text{pressureCook}(x, y) \vee a = \text{fry}(x, y) \vee$
 $a = \text{deepFry}(x, y) \vee a = \text{stir}(x, y)$
 $\vee \text{cooked}(x, s).$

Correctness of the New BATs

- **Theorem 1:**
For any modular BAT $\mathcal{D}^{\mathcal{H}}$, there exists an equivalent \mathcal{D} of Reiter's BAT format, where equivalence means that for any FO regressable sentence W that has no occurrence of sp (or sp^*), we have

$$\mathcal{D}^{\mathcal{H}} \models W \text{ iff } \mathcal{D} \models W.$$

- **Theorem 2 (regression theorem):**

$$\mathcal{D}^{\mathcal{H}} \models W \text{ iff } \mathcal{D}_{\text{SO}}^{\mathcal{H}} \cup \mathcal{D}_{\text{una}} \models \mathcal{R}[W].$$

- An extended regression operator is defined to handle sp^*

- **Theorem 3:**

Although the formal definition of sp^* is second-order, the reasoning in $\mathcal{D}^{\mathcal{H}}$ can be reduced to a FOL reasoning only.

Computational Advantages

- **Theorem 4 (Exponential speed-up):**

If the sub-graph rooted at action A in the digraph of an action hierarchy is a complete k -ary tree ($k \geq 2$) with n action functions as its nodes, the computational complexity of extended regression on $sp^*(a, A(t))$ is $\Theta(\log_k n)$, while the computational complexity of Reiter's regression on an equivalent replacement is $\Theta(n)$.

Computational Advantages (Example)

- An Example: regression on $\text{cooked}(\text{Egg}, \text{do}(\text{deepFry}(\text{Egg}, \text{FryingPan}), S_0))$

Remind

□Modular BAT

$$\text{cooked}(x, \text{do}(a, s)) \equiv \text{sp}^*(a, \text{cook}(x)) \vee \text{cooked}(x, s).$$

□Reiter's BAT

$$\text{cooked}(x, \text{do}(a, s)) \equiv$$

$$a = \text{cook}(x) \vee a = \text{lowOilCook}(x) \vee a = \text{oilyCook}(x) \vee$$

$$a = \text{steam}(x, y) \vee a = \text{boil}(x, y) \vee a = \text{stew}(x, y) \vee$$

$$a = \text{broil}(x, y) \vee a = \text{bake}(x, y) \vee a = \text{ovenCook}(x, y) \vee$$

$$a = \text{roast}(x, y) \vee a = \text{pressureCook}(x, y) \vee a = \text{fry}(x, y) \vee$$

$$a = \text{deepFry}(x, y) \vee a = \text{stir}(x, y)$$

$$\vee \text{cooked}(x, s).$$

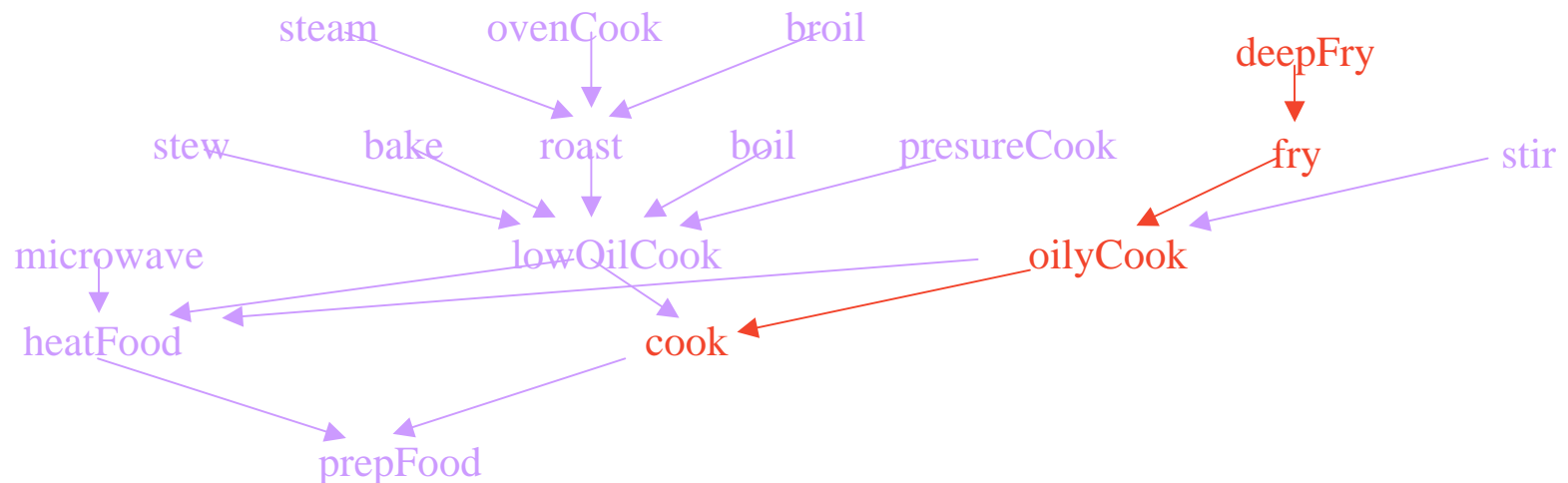
Computational Advantages (Example)

- With modular BAT

Extended regression on $sp^*(\text{deepFry}(\text{Egg}, \text{FryingPan}), \text{cook}(\text{Egg}))$

= 1 step of regression on $sp^*(\text{deepFry}(\text{Egg}, \text{FryingPan}), \text{cook}(\text{Egg}))$

+ 3 steps of checking whether $sp^*(\text{deepFry}(\text{Egg}, \text{FryingPan}), \text{cook}(\text{Egg}))$ is true



Computational Advantages (Example)

- With Reiter's BAT

Regression on equivalent part of $sp^*(\text{deepFry}(\text{Egg}, \text{FryingPan}), \text{cook}(\text{Egg}))$, i.e.,

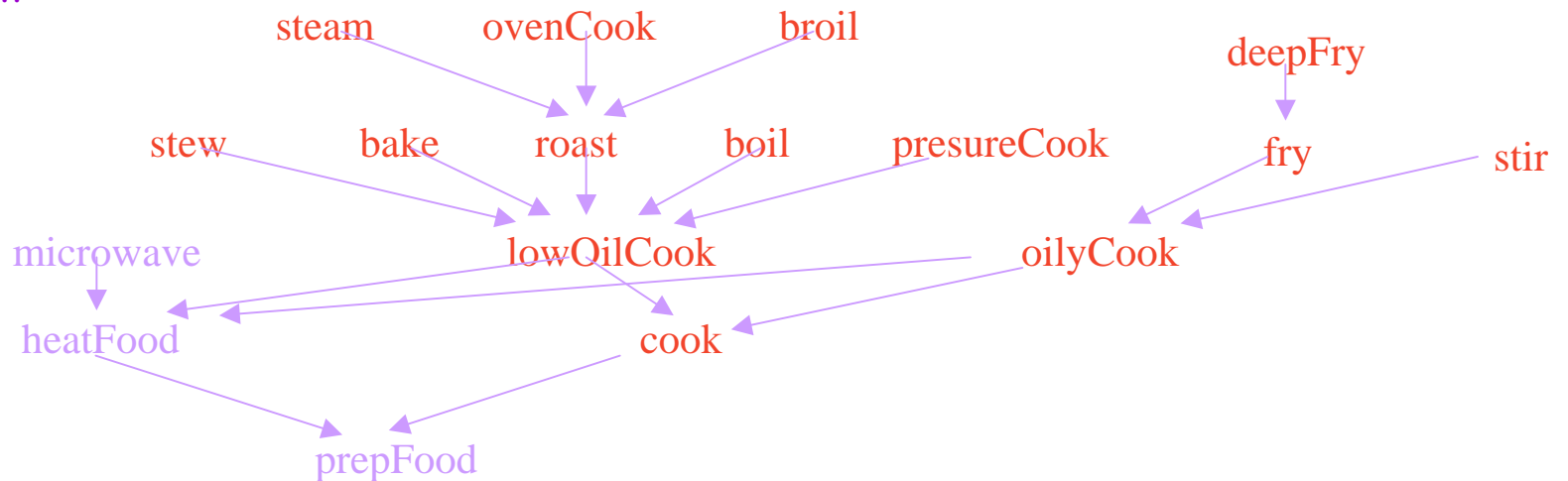
$\mathcal{R}[\text{deepFry}(\text{Egg}, \text{FryingPan}) = \text{cook}(\text{Egg}) \vee \dots$

$\vee \text{deepFry}(\text{Egg}, \text{FryingPan}) = \text{stir}(\text{Egg}, \text{FryingPan})]$ (totally 14 disjunctions of actions)

$= \mathcal{R}[\text{deepFry}(\text{Egg}, \text{FryingPan}) = \text{cook}(\text{Egg})] \vee$

$\mathcal{R}[\text{deepFry}(\text{Egg}, \text{FryingPan}) = \text{cook}(\text{Egg}) \vee \dots \vee \text{deepFry}(\text{Egg}, \text{FryingPan}) = \text{stir}(\text{Egg}, \text{FryingPan})]$

$= \dots \dots$



Conclusion and Future Work

- A theory of handling large taxonomies of action in the situation calculus
- Gain both representational and computational advantages
- Related research
 - A modular action description language [Lifschitz & Ren 2006]
 - Logical aspects of events: Quantification, sorts, composition and disjointness [Kaneiwa & Tojo 2005]
 - Taxonomic plan reasoning [Devanbu & Litman 1996]
 - Generalized plan recognition [Kautz & Allen 1986, Kautz 1991]
 - (De)Composition of Situation Calculus Theories [Amir 2000]
- Future work
 - Hierarchies of complex actions
 - Automatic construction of action hierarchies (under certain restrictions)
 - Combining with Amir's decomposed situation calculus theories
 - Etc.

The End

Thank you!