Reasoning about Large Taxonomies of Actions in the Situation Calculus

Yilan Gu
Department of Computer Science, University of Toronto, Canada
Mikhail Soutchanski
Department of Computer Science, Ryerson University, Canada

Motivation

The situation calculus [Reiter, 2001]

- Initially proposed by J. McCarthy, enriched by R. Reiter et al at Univ. of Toronto
- A high-level first-order logic language
- Representing and reasoning about actions in a natural and compact way

Problem: Handling effects of action functions individually and impractical to handle large scales of action functions

Our Idea

- Using common-sense knowledge: Considering one action as a specialization of another
- E.g., traveling and shipping can be considered as specializations of moving

Summary of Contributions

1. Representing large taxonomies of actions in a hierarchical way
2. Representing effects of actions much more compactly based on action hierarchies
3. Reasoning problems can be solved exponentially faster (sometimes)
4. Bringing possible knowledge engineering advantages

Background: The Situation Calculus

Sorts of the language

- *Action functions:* $fry(x,y)$ (frying x using y), $wash(y)$ (washing y)
- *Situations:* S_0 (the initial situation), $do(a,s)$ (the situation after doing a in s)
- *Objects:* persons, locations, food items, ...

Fluents

- System features whose truth values may vary from situation to situation
- Predicates or functions with the last argument of sort situation
  E.g., $FoodReady(x,s)$ – food x is ready in the situation s
  $Dirty(x,s)$ – x is dirty in the situation s

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

- Describing actions and their effects in a dynamic system
- A set of first-order axioms, mainly includes:
  $\mathcal{D}_{ap}$ (precondition axioms): For each $A(\vec{x})$, $Poss(A(\vec{x}),s) \equiv \phi_A(\vec{x},s)$
  $\mathcal{D}_{ssa}$ (successor state axioms): For each fluent $F(\vec{y},s)$,
  $$F(\vec{y},do(a,s)) \equiv \bigvee_i \psi_i^+(\vec{y},a,s) \vee F(\vec{y},s) \wedge \neg(\bigvee_j \psi_j^-(\vec{y},a,s))$$
  $\mathcal{D}_{S_0}$ (an initial theory): All facts hold in the initial situation S_0
  $\mathcal{D}_{una}$ (unique name axioms for action names) and Σ (foundational axioms)

An example of kitchen activity domain

- Considering actions involved in a kitchen such as $cook(x)$, $fry(x,y)$, $wash(x)$, ..., etc, and fluents such as $FoodReady(x,s)$, $Dirty(x,s)$, ..., etc. (See actions in the digraph on the right hand-side)
- An example of the precondition axioms:
  $$Poss(wash(x),s) \equiv WashableObject(x) \wedge Dirty(x,s)$$
- An example of the successor state axioms:
  $$\begin{aligned} &Dirty(y,do(a,s)) \equiv \\ &\quad (\exists x)a = steam(x,y) \vee (\exists x)a = boil(x,y) \vee (\exists x)a = pressureCook(x,y) \vee \\ &\quad (\exists x)a = broil(x,y) \vee (\exists x)a = bake(x,y) \vee (\exists x)a = ovenCook(x,y) \vee \\ &\quad (\exists x)a = roast(x,y) \vee (\exists x)a = deepFry(x,y) \vee (\exists x)a = stir(x,y) \vee \\ &\quad (\exists x)a = fry(x,y) \vee (\exists x)a = stew(x,y) \vee (\exists x)a = makeSalad(x,y) \vee \\ &\quad (\exists x)a = prepHotDr(x,y) \vee (\exists x)a = prepColdDr(x,y) \vee \\ &\quad Dirty(y,s) \wedge \neg[a = wash(y) \vee a = handWash(y)]. \end{aligned}$$
- Some examples of the axioms or facts in the initial theory:
  $Food(Egg_0)$, $Vessel(Pot_0)$, ..., $(\forall x)Vessel(x) \supset \neg Dirty(x,S_0)$, $(\forall x)Food(x) \supset \neg FoodReady(x,S_0)$, ..., etc.

Reasoning using regression operator R

- Defined recursively using basic action theories
  E.g., $R[W_1 \vee W_2] = R[W_1] \vee R[W_2]$, $R[(\exists x)W] = (\exists x)R[W]$, ...,
  $R[F(\vec{y},do(a,s))] = R[\phi_F(\vec{y},a,s)]$ where $F(\vec{y},do(a,s)) \equiv \phi_F(\vec{y},a,s)$ using $\mathcal{D}_{ssa}$
- Reiter's regression theorem: $\mathcal{D} \models W$ iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models R[W]$

Action Hierarchies

An action diagram $\mathcal{H}$

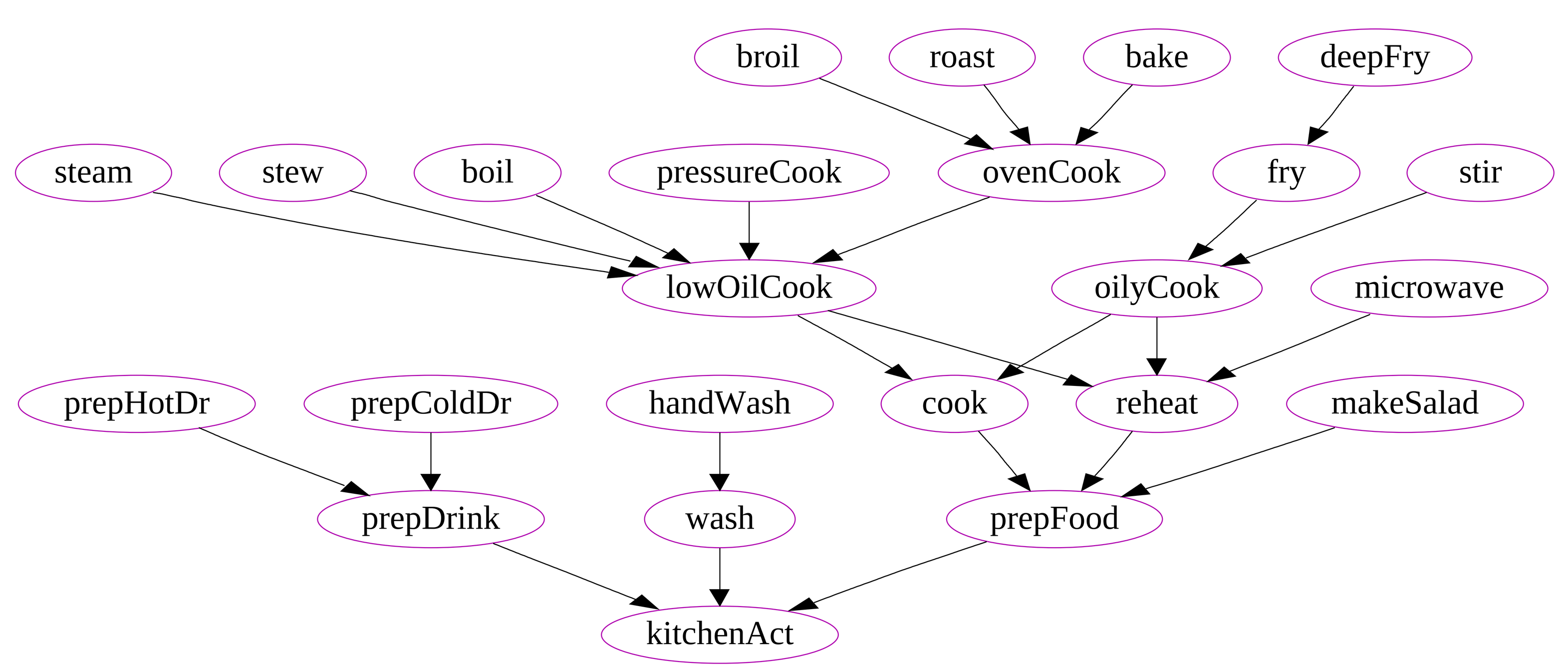
- $sp(a_1,a_2)$: Action a_1 is a (direct) specialization of action a_2
- A collection of finitely many axioms: $sp(A_1(\vec{x}), A_2(\vec{y})) \equiv \phi_{A_1,A_2}(\vec{x},\vec{y})$
  E.g., $sp(fry(x,y), oilyCook(x))$, $sp(oilyCook(x), cook(x))$, ..., etc
- Acyclic, antisymmetric, multiple inheritance

An action hierarchy $\mathcal{H}^\star = \mathcal{H}+$**the definition of** isA

- $isA(a_1,a_2)$: Action a_1 is a (distant) specialization of action a_2
- The reflexive-transitive closure of sp defined in second-order logic
- Theorem:
  For any action functions $A_1(\vec{x})$ and $A_2(\vec{y})$, $\mathcal{H}^\star \models isA(A_1(\vec{x}),A_2(\vec{y})) \equiv \psi_{A_1,A_2}(\vec{x},\vec{y})$, where $\psi_{A_1,A_2}(\vec{x},\vec{y})$ is a first-order formula and can be found in finitely many steps.

Event slots

- $EventSlot(a,x,l)$: x is an event slot l in action a
- Identifying the relationship between an action and each of its argument
  E.g., $EventSlot(fry(x,y),x,ActTarget)$, $EventSlot(fry(x,y),y,Utensil)$, etc.



A Modular Basic Action Theory $\mathcal{D}^H$

Format of axioms

- $\mathcal{D}_{ap}$, $\mathcal{D}_{una}$ and Σ are as usual
- $\mathcal{D}^H_{S_0}$: includes action hierarchy axioms $\mathcal{H}^\star$ and axioms for event slots in addition
- $\mathcal{D}^H_{ssa}$: New representation of basic action theories using isA
  E.g., $Dirty(y,do(a,s)) \equiv (\exists x)isA(a,prepFood(x)) \wedge EventSlot(a,y,Utensil) \vee$
  $(\exists x)isA(a,prepDrink(x)) \wedge EventSlot(a,y,Utensil) \vee$
  $Dirty(y,s) \wedge \neg isA(a,wash(y))$

Regression: operator is extended to handle isA and $EventSlot$

Correctness of Modular Basic Action Theories

1. For any modular BAT $\mathcal{D}^H$, there exists an equivalent $\mathcal{D}$ of Reiter's BAT format, where equivalence means that for any first-order regressable sentence W, $\mathcal{D}^H \models W$ iff $\mathcal{D} \models W$.
2. A regression theorem similar to Reiter's regression theorem is proved: $\mathcal{D}^H \models W$ iff $\mathcal{D}^H_{S_0} \cup \mathcal{D}_{una} \models R[W]$.
3. Although the formal definition of isA is second-order, the reasoning in $\mathcal{D}^H$ can be reduced to a first-order logic reasoning only.

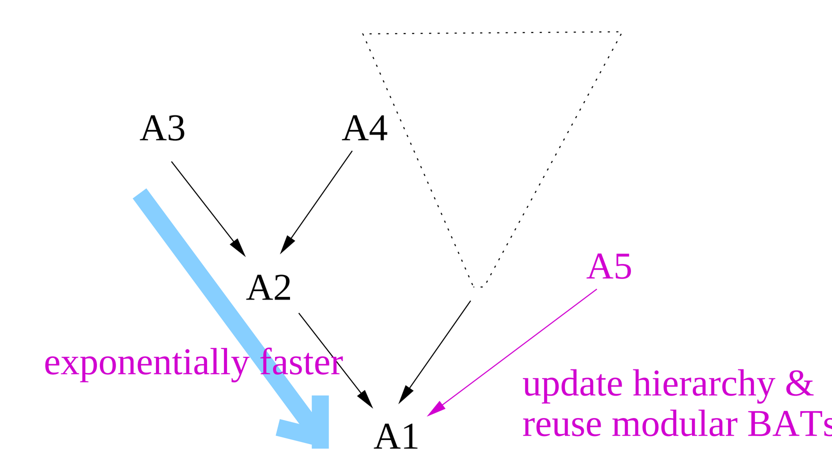
Advantages of The New Approach

Computational advantages

When $\mathcal{H}$ has a tree or forest structure, then the reasoning on isA is exponentially faster than the reasoning on its equivalent clause in Reiter's BAT representation.

Knowledge engineering advantages

- Allowing taxonomic reasoning about event slot hierarchies and action hierarchies
- Possible applications to action/service retrieval
- Easiness for system update and reuse of action hierarchies and modular BATs



Implementation (in progress): E-business domain and kitchen activity domain

References and Most Related Work

[Amir, 2000] (De)Composition of Situation Calculus Theories
[Barker, Porter & Clark, 2001] A Library of Generic Concepts for Composing KBs
[Gil, 2005] Description Logics and Planning
[Gu & Soutchanski, 2007] Modular Basic Action Theories
[Lifschitz & Ren, 2006] A Modular Action Description Language
[Reiter, 2001] Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems.