Action Theories over Generalized Databases with Equality Constraints

Fabio Patrizi

Sapienza Università di Roma, Italy
patrizi@dis.uniroma1.it

Ongoing joint work with Stavros Vassos

Bolzano – April 12, 2014

Situation Calculus

[McCarthy63,McCarthyHayes69,Reiter01]

First-order multi-sorted language for reasoning about actions

Sorts

- Objects Δ : (possibly infinite) domain of discourse –$block_1, block_2, \ldots$
- Actions Act (defined using finite set $\mathcal{A}$ of action function symbols):
  - finitely many action types –$pick(x)$, $stack(x, y)$
  - possibly infinitely many actions –$pick(block_1), pick(block_2), \ldots$
- Situations $\mathcal{S}$: world histories (defined inductively)
  - S_0: constant denoting initial situation
  - $do(s, \alpha)$ situation resulting from executing (ground) action α at s

Fluents

- predicates asserting properties of objects in situations –$On(x, y, s)$
- NO functional fluents (here)

Basic Action Theories (BATs)

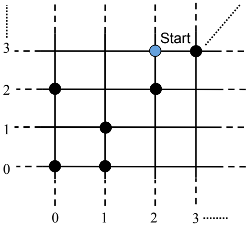
$\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \Sigma$

- Initial situation description $\mathcal{D}_0$:
  FO axioms (*uniform* in S_0) defining initial configuration
- Precondition axioms $\mathcal{D}_{ap}$ –when actions are executable:
  $Poss(A(\vec{x}), s) \equiv \Phi_A(\vec{x}, s)$ (FO)
- Successor state axioms $\mathcal{D}_{ss}$ –action effects:
  $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$ (FO)
- Uniqueness of action names $\mathcal{D}_{una}$:
  $A(\vec{x}) \neq A'(\vec{y})$, $A(\vec{x}) = A(\vec{y}) \supset \vec{x} = \vec{y}$ (FO)
- Foundational axioms Σ:
  (domain-independent) formal definitions of
  - *situation tree* (SO, due to induction)
  - ordering $\sqsubseteq$ among situations

We restrict to *standard interpretations* [Levesque98]: named objects; u.n.a. axioms for constants; axioms for equality ($\mathcal{E}$)

Basic Action Theories

Example



- Infinite grid
- Start in $(2, 3)$
- Can move only along lines
- Can change direction only by stopping on marked crossings

Basic Action Theories

Example

- Action types: $\mathcal{A} = \{moveTo(x, y)\}$
- Fluents: $\mathcal{F} = \{At(x, y, s), Dest(x, y, s), Cross(x, y, s)\}$
- $\mathcal{D}_0$:
  - $At(x, y, S_0) \equiv x = 2 \wedge y = 3$
  - $Dest(x, y, S_0) \equiv x = 2$
  - $Cross(x, y, S_0) \equiv (x = y) \vee (x = 0 \wedge y = 2) \vee (x = 1 \wedge y = 0)$
- $\mathcal{D}_{ap}$:
  - $Poss(moveTo(x, y), s) \equiv Dest(x, y, s)$
- $\mathcal{D}_{ss}$:
  - $Cross(x, y, do(moveTo(x', y'), s)) \equiv Cross(x, y, s)$
  - $At(x, y, do(moveTo(x', y'), s)) \equiv (x = x' \wedge y = y')$
  - $Dest(x, y, do(moveTo(x', y'), s)) \equiv (Cross(x', y', s) \wedge (x = x' \vee y = y'))\vee$ $\exists x'', y''. At(x'', y'', s) \wedge [(x' = x'' \wedge y' \neq y'' \wedge x = x')\vee$ $(y' = y'' \wedge x' \neq x'' \wedge y = y') \vee (y' = y'' \wedge x' = x'' \wedge Dest(x, y, s))]$

OBS: $Dest$ extension can be infinite

Situation Calculus

Reasoning Tasks

Regression[PirriReiter99] (not this work) Reduce reasoning about a future situation to reasoning about initial situation (weakest precondition)

Progression[LinReiter97] (this work) Provide a "complete" description $\mathcal{D}_\alpha$ of the new configuration obtained by executing α in S_0

Projection[Reiter01] (this work) Predict whether a condition $\phi(s)$ holds after a sequence $\alpha_0, \ldots, \alpha_n$ of actions is executed (we consider a more general form)

Progression

Progression: Set of sentences $\mathcal{D}_\alpha$ such that $\mathcal{D}$ and $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_\alpha$ "coincide" from $do(\alpha, S_0)$ on

Important questions:

- Is progression, i.e. $\mathcal{D}_\alpha$, FO-definable?
- (When) Can we come up with a FO $\mathcal{D}_\alpha$?

Some answers:

- Second-Order $\mathcal{D}_\alpha$ required [LinReiter97, VassosLevesque13]
- Practically-relevant cases exist of FO-progressable theories [LinReiter97]:
  - Initial KB is *definitional* (in our case a possibly infinite database):

$$\mathcal{D}_0 = \{ \bigwedge_{F \in \mathcal{F}} \forall \vec{x}. F(\vec{x}, S_0) \equiv \phi_F(\vec{x}) \}, \phi_F \text{ mentions no situation}$$

  - Context-free SSAs: F depends only on F at previous situation

Projection

(Simple) Projection: given a sequence of actions $\alpha_1 \cdots \alpha_n$ check whether $\mathcal{D} \models \phi(s)$, for $s = do(\alpha_n, \ldots, do(\alpha_1, S_0)))$

Through regression, projection can be reduced to a query over the initial KB (to answer which, theorem proving is needed in general)

Decidable (and practical) in few cases:

- initial KB is a regular database[Reiter92]
- incomplete knowledge as proper KB + local effects [LiuLevesque05] (sometimes complete)
- two-variable fragment of FO [GuSoutchanski07]
- bounded action theories [DeGiacomo-etal12] (beyond projection)

Progression and Projection

Action sequence: $\alpha_1 \cdots \alpha_n$, Property: ϕ

Progression can be used as a basic step for projection:

1. Start with $\mathcal{D}_0$ and $\alpha = \alpha_1$
2. Progress current $\mathcal{D}_0$ w.r.t. current action α, getting $\mathcal{D}_\alpha$
3. Update $\mathcal{D}_0$ with obtained progression, i.e., let $\mathcal{D}_0 = \mathcal{D}_\alpha$
4. Iterate 2 with $\alpha = \alpha_{i+1}$ until $i = n$
5. Check whether obtained $\mathcal{D}_\alpha$ satisfies $\phi(s)$

$$\mathcal{D}_0 \xrightarrow{\alpha_1} \mathcal{D}_{\alpha_1} \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_{n-1}} \mathcal{D}_{\alpha_{n-1}} \xrightarrow{\alpha_n} \mathcal{D}_{\alpha_n}$$

$$\mathcal{D}_{\alpha_n} \models \phi?$$

Decidable and practical when progression and $\models$ are so

BATs with Definitional Initial KB: Progression

[LinReiter97]

Progression obtained by syntactically replacing fluent atoms with their definition in $\mathcal{D}_0$

For each SSA $F(\vec{x}, do(a, s)) \equiv \Phi(\vec{x}, a, s)$:

- Replace every atom $F_j(\vec{o}, s)$ in $\Phi(\vec{x}, a, s)$ with the definition ϕ_j of F_j in $\mathcal{D}_0$

The obtained set $\mathcal{D}_\alpha$ is a progression

NOTE: at every progression step, size of axioms in $\mathcal{D}_\alpha$ grows

$\mathcal{D}_\alpha$: set of (FO) axioms $\rightarrow$ query answering needs theorem proving

We look for a more practical, *ready-to-use* form of progression

BATs with Definitional KB: Progression

Example

- $s = do(moveTo(2,4), S_0)$
- $\mathcal{D}_0 = \{At(x,y,S_0) \equiv x = 2 \wedge y = 3, Dest(x,y,S_0) \equiv x = 2,$
  $Cross(x,y,S_0) \equiv (x = y) \vee (x = 0 \wedge y = 2) \vee (x = 1 \wedge y = 0)\}$
- $\mathcal{D}_{ss} = \{Cross(x,y,do(moveTo(x',y'),s)) \equiv Cross(x,y,s),$
  $At(x,y,do(moveTo(x',y'),s)) \equiv (x = x' \wedge y = y'),$
  $Dest(x,y,do(moveTo(x',y'),s)) \equiv (Cross(x',y',s) \wedge (x = x' \vee y = y'))\vee$
  $\exists x'', y''.At(x'',y'',s) \wedge [(x' = x'' \wedge y' \neq y'' \wedge x = x')\vee$
  $(y' = y'' \wedge x' \neq x'' \wedge y = y') \vee (y' = y'' \wedge x' = x'' \wedge Dest(x,y,s))]\}$
- $\mathcal{D}_\alpha = \{Dest(x,y,do(moveTo(2,4),S_0)) \equiv$
  $(((2 = 4) \vee (2 = 0 \wedge 4 = 2) \vee (2 = 1 \wedge 4 = 0)) \wedge (x = 2 \vee y = 4))\vee$
  $\exists x'', y''.x'' = 2 \wedge y'' = 3 \wedge [(2 = x'' \wedge 4 \neq y'' \wedge x = 2)\vee$
  $(4 = y'' \wedge 2 \neq x'' \wedge y = 4) \vee (4 = y'' \wedge 2 = x'' \wedge x = 2)], \ldots\}$

Generalized Databases (with Equality Constraints)

[Kanellakis-etal95]

Generalized k-tuple : (models of) conjunction of equality constraints involving k variables $x_1, \ldots, x_k$

Generalized relation (of arity k) : (model of) disjunction of k-tuples over $x_1, \ldots, x_k$

Generalized Database: set of (models of) generalized relations

Example (Generalized relation)

$Cross(x, y, S_0) \equiv (x = y) \vee (x = 0 \wedge y = 2) \vee (x = 1 \wedge y = 0)$

Answering Queries over Generalized DBs

Generalized relations can be infinite: not representable extensionally

What if we need to answer queries on a generalized DB?

Theorem (Kanellakis-etal95)

Query answers on Generalized DBs are computable and representable as generalized relations:

1. *Replace each relation atom R_i in the query ϕ by its formula ϕ_R*
2. *Build all the (finitely many, up to isomorphism) generalized tuples of appropriate arity*
3. *Keep only the tuples consistent with the new query ϕ' obtained in 1*

Corollary: logical equivalence (under $\mathcal{E}$) is decidable

Thus:

- Effective procedure to answer queries on a class of infinite DBs
- A closed representation system

We exploit these features to address progression and projection

BATs with Generalized Fluent DBs

Definition

Generalized fluent DB (GFDB) $\mathcal{D}_0$:

$$\{ \bigwedge_{F_i \in \mathcal{F}} \forall \vec{x}_i . F(\vec{x}_i, S_0) \equiv \phi_i(\vec{x}_i) \}$$

with $\phi_i(\vec{x}_i)$ a generalized relation formula (with equality constraints)

Intuition: extension of each fluent as a generalized relation

Definition

A BAT-GFDB $\mathcal{D}$ is a BAT s.t. $\mathcal{D}_0$ is a GFDB

BAT-GFDBs and BATs with Definitional KB

Theorem

For any definitional KB there exists an equivalent generalized fluent database, and viceversa.

From definitional KB to GFDB (viceversa obvious):

- *Eliminate quantifiers (FO theories of equality admit quantifier elimination)*
- *Rewrite as DNF*

Constructive: actual procedure to transform a definitional KB into a GFDB

Progression of BAT-GFDBs

Progression as query answering on (generalized) DBs

Example

- $Dest(x, y, do(moveTo(x', y'), s)) \equiv (Cross(x', y', s) \wedge (x = x' \vee y = y'))\vee$
  $\exists x'', y''.At(x'', y'', s) \wedge [(x' = x'' \wedge y' \neq y'' \wedge x = x')\vee$
  $(y' = y'' \wedge x' \neq x'' \wedge y = y') \vee (y' = y'' \wedge x' = x'' \wedge Dest(x, y, s))]$

  $Dest(x, y, do(moveTo(2, 4), S_0))$ can be obtained by answering the RHS above on $\mathcal{D}_0$

- $\mathcal{D}_\alpha = \{Dest(x, y, do(moveTo(2, 4), S_0)) \equiv (x = 2), \ldots\}$

$\mathcal{D}_\alpha$ is now more of a *materialized update* than a logical specification!

Theorem

There always exists a progression $\mathcal{D}_\alpha$ that is a GFDB

Simple Projection Over BAT-GFDBs

We can iteratively progress a theory w.r.t. a sequence of actions $\alpha_1, \ldots, \alpha_n$, obtaining a GFDB at every step:

$$\mathcal{D}_0 \xrightarrow{\alpha_1} \mathcal{D}_{\alpha_1} \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} \mathcal{D}_{\alpha_n}$$

So, we can check whether $\mathcal{D} \models \phi(do(\alpha_n, \ldots, do(\alpha_1, S_0)))$ by simply checking whether $\mathcal{D}_{\alpha_n} \models \phi(do(\alpha_n, \ldots, do(\alpha_1, S_0)))$ (recall ϕ is local)

NOTE: Decidability of projection for definitional KBs was known and based on regression. When a method is preferable needs further investigation.

Generalized Projection Over BAT-GFDBs

Generalization: ϕ may refer to any number of future situations

$$\phi = \forall s.do(moveTo(2,4), S_0) \sqsubseteq s \supset \exists x, y.Dest(x, y, s)$$

(After executing $moveTo(2,4)$, any future situation allows for at least one destination)

Language $\mathcal{L}_p$ of generalized projection queries:

$$\phi := x = c \mid x = y \mid F(\vec{x}, s) \mid F(\vec{x}, \sigma) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x.\phi$$

$$\varphi := \phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists s.\sigma \sqsubseteq s \wedge \varphi$$

where ϕ is uniform in s or in σ, with free variables only of sort situation

We consider only sentences in $\mathcal{L}_p$

Generalized Projection Over BAT-GFDBs

BAT-GFDBs in general infinite-state $\rightarrow$ cannot simply "visit" the model of $\mathcal{D}$ to check whether $\mathcal{D} \models \phi$

However: for a special class of BAT-GFDBs we can reduce the check to one over a finite structure

Constant-bounded BAT-GFDBs

Definition

A BAT-GFDB $\mathcal{D}$ is C-bounded by B if the state of every executable situation can be represented as a GFDB mentioning at most B distinct constants.

NOTE: Semantic definition. Syntactic conditions to be investigated.

Example

The grid theory is C-bounded. At every situation, we need:

- 2 constants for current position (At)
- 3 constants for $Cross$
- At most 2 constants for $Dest$

(Recall we have named objects)

C-bounded BAT-GFDBs generalize bounded BATs [DeGiacomo-etal12]

Generalized Projection over C-bounded BAT-GFDBs

Theorem

Checking whether $\mathcal{D} \models \varphi$ for a C-bounded BAT-GFDB and a generalized projection query is decidable.

Crux of the proof:

- Base case of projection queries is local (FO) sentence
- For given B, only finitely many equivalence classes of *logically equivalent (under $\mathcal{E}$)* GFDBs exist
- Only equivalence class matters for the base case

Can build a *finite-state* TS $\hat{T}_{\mathcal{D},\varphi}$ with isomorphism types as states, that preserves transitions between types

Generalized Projection over C-bounded BAT-GFDBs

Construction of $\hat{T}_{\mathcal{D},\varphi}$

Given: a BAT-GFDB $\mathcal{D}$ and a generalized projection query $\varphi \in \mathcal{L}_p$:

1. Fix a *finite* set of constants H containing:
   - all constants mentioned in $\mathcal{D}$ ($\mathcal{C}_{\mathcal{D}}$) and φ ($\mathcal{C}_{\varphi}$)
   - $B \times |\mathcal{F}|$ fresh constants
   - $N_{\mathcal{A}}$ fresh constants, with $N_{\mathcal{A}}$ max num of parameters in action types
2. From $\mathcal{D}_0$, iteratively progress the current situation
   - Consider all possible actions for $\mathcal{A}$ and H
   - Generate progression in the form of BAT-GFDB
   - Record progression steps as action-labelled transitions
   - If a logically equivalent progression has been generated, reuse it (i.e., *connect back*)

   Stop when all progressions (up to logical equivalence) are expanded

OBS: by finiteness of H, $\hat{T}_{\mathcal{D},\varphi}$ is finite-state

Generalized Projection over C-bounded BAT-GFDBs

Property Check

Theorem

$$\mathcal{D} \models \varphi \textit{ iff } \hat{T}_{\mathcal{D},\varphi} \models \varphi$$

We can check φ against the finite $\hat{T}_{\mathcal{D},\varphi}$ instead of the infinite model of $\mathcal{D}$

Can be easily done using a MC-like procedure

Conclusions

1. Generalized DBs characterize definitional KBs (without non-fluent predicates) and generalize bounded BATs
2. Transformation from definitional KB to GFDB provided
3. Closed representation system
4. Progression of GFDBs closer to an *actual update* than logical specification
5. For BAT-GFDBs, standard projection and a generalized form decidable (actual procedure given)
6. To date most expressive SitCalc theory with infinite fluent extensions and decidable progression and generalized projection

Future Work

1. Investigate syntactical conditions that guarantee C-boundedness
2. Add forms of incomplete knowledge (e.g., bounded unknowns [VassosP13])
3. Consider special non-GFDB-expressible fluents such as linear order
4. Exploit results for actual implementation on Golog family of high-level agent programming languages