The Nondeterministic Situation Calculus

Yves Lespérance

Sapienza University of Rome, April 2024

Outline

Outline

The Nondeterministic Situation Calculus - Motivation

Motivation

- In standard Situation Calculus (SitCalc), atomic actions are deterministic
- The situation resulting from doing action a in situation s is denoted by the term $do(a, s)$ – unique in each model.
- But often need to represent actions that are **nondeterministic** and have different possible outcomes that are not under agent's control, e.g., flipping a coin, attempting to stack a block, etc.
- Many previous approaches, e.g., using nondeterministic programs in Golog
- But most are cumbersome and/or don't clearly distinguish between choices that can be made by the agent and choices made by the environment

The Nondeterministic Situation Calculus - Motivation

Nondeterministic Situation Calculus [DL21]

- Here, we present a **simple extension to the standard SitCalc that accommodates nondeterministic actions**
- Preserves Reiter's solution to the frame problem and answering projection queries through regression
- We also show how to handle FOND planning and ConGolog high-level program execution in nondeterministic domains

Outline

Nondeterministic Situation Calculus – The Approach

The Approach

- Outcome of a nondeterministic action is determined by the agent action and the environment's reaction
- Every action type/function $A(\vec{x}, e)$ takes an additional **environment reaction** parameter e ranging over new sort Reaction
- Call this $A(\vec{x}, e)$ a **system action**
- Agent performs the reaction-suppressed version of the action $A(\vec{x})$, i.e., the **agent action**
- The environment chooses the **reaction** e
- This allows us to quantify separately over agent actions and environment reactions
- We clearly distinguish between the nondeterminism associated with **agent choices** which is **angelic** and that associated with **environment choices** which is **devilish**

Outline

Nondeterministic Basic Action Theories (NDBATs)

NDBATs

A Nondeterministic Basic Action Theory (NDBAT) is a BAT where:

- every action function takes an environment reaction parameter
- for each agent action, we have an **agent action precondition formula**, stating when the action can be performed by the agent:

$$Poss_{ag}(A(\vec{x}), s) \doteq \phi_A^{agPoss}(\vec{x}, s)$$

- we have a **reaction independence requirement**: the precondition for agent action must be independent of any environment reaction:

$$\forall e.Poss(A(\vec{x}, e), s) \supset Poss_{ag}(A(\vec{x}), s)$$

- we have a **reaction existence requirement**: if the precondition of the agent action holds then there exists a reaction which makes the complete system action executable, i.e., the environment can't prevent agent from performing it if its agent action precondition holds:

$$Poss_{ag}(A(\vec{x}), s) \supset \exists e.Poss(A(\vec{x}, e), s)$$

- these requirements **must be entailed** by the action theory for it to be an NDBAT

Nondeterministic Basic Action Theories (NDBATs)

NDBATs (cont.)

As usual, we have:

- (system) action precondition axioms, which specify the possible environment reactions
- successor state axioms, which specify how fluents are affected by the system actions
- initial situation axioms, which describe the initial conditions
- unique names axioms for actions and foundational axioms

Example NDBAT for Track World Domain $\mathcal{D}_{trk}$

E.g. NDBAT $\mathcal{D}_{trk}$

Have an agent can move back and forth along a track.

Have nondeterministic agent action fwd which moves her forward by one or two positions.
Also have deterministic action $bk1$ to go backwards by one postion.

$$Poss_{ag}(fwd, s) \doteq True$$
$$Poss_{ag}(bk1, s) \doteq pos(s) > 0$$
$$Poss(fwd(e), s) \equiv Poss_{ag}(fwd, s) \wedge (e = f(1) \vee e = f(2))$$
$$Poss(bk1(e), s) \equiv Poss_{ag}(bk1, s) \wedge e = f(1)$$
$$pos(do(a, s)) = k \equiv$$

fluent's value is unbounded!

$$\begin{array}{l} a = fwd(e) \wedge \exists n.e = f(n) \wedge k = pos(s) + n \vee \\ a = bk1(e) \wedge k = pos(s) - 1 \vee \\ pos(s) = k \wedge (\neg\exists e.a = fwd(e)) \wedge (\neg\exists e.a = bk1(e)) \end{array}$$
$$pos(S_0) = 0$$

Outline

Action Execution(s) in NDsitCalc

In standard sitCalc/Golog, we have:

$$Do(A(\vec{x}, e), s, s') \doteq Poss(A(\vec{x}, e), s) \wedge s' = do(A(\vec{x}, e), s)$$

Deterministic! Still holds in NDsitCalc for system actions.

But an **agent action may have several executions depending on the environment reaction**.

Definition $Do_{ag}(\sigma, s, s')$

i.e., the system **may reach situation** s' when the agent executes **agent action sequence** σ starting in situation s:

$$Do_{ag}(\epsilon, s, s') \doteq s = s'$$
$$Do_{ag}([A(\vec{x}), \sigma], s, s') \doteq \exists e. Poss(A(\vec{x}, e), s) \wedge Do_{ag}(\sigma, do(A(\vec{x}, e), s), s')$$

Agent Action Execution in Track World Example

Example

For our running example $\mathcal{D}_{trk}$, we have:

$$\begin{array}{l}\mathcal{D}_{trk} \models pos(do(fwd(f(1)), S_0)) = 1 \wedge \\ \quad pos(do(fwd(f(2)), S_0)) = 2 \wedge \\ \quad pos(do(bk1(f(1)), do(fwd(f(2)), S_0))) = 1 \wedge \\ \quad (Do_{ag}(fwd, S_0, s) \equiv \\ \qquad s = do(fwd(f(1)), S_0) \wedge pos(s) = 1 \vee \\ \qquad s = do(fwd(f(2)), S_0) \wedge pos(s) = 2) \wedge \\ \quad (Do_{ag}(bk1, S_0, s) \equiv s = do(bk1(f(1)), S_0))\end{array}$$

Projection in NDsitCalc

Agent actions' outcomes depend on environment reactions.

Definitions

$CertainlyAfter(\sigma, \phi, s)$ means that ϕ is **certainly true after** the sequence of agent actions σ

$$CertainlyAfter(\sigma, \phi, s) \doteq \forall s'.Do_{ag}(\sigma, s, s') \supset \phi[s']$$

$PossiblyAfter(\sigma, \phi, s)$ means that ϕ is **possibly true after** the sequence of agent actions σ

$$PossiblyAfter(\sigma, \phi, s) \doteq \exists s'.Do_{ag}(\sigma, s, s') \wedge \phi[s']$$

Projection in NDsitCalc in Track World Example

Example

For our running example $\mathcal{D}_{trk}$, we have:

$$\begin{aligned}\mathcal{D}_{trk} \models\ & PossiblyAfter([fwd, bk1], (pos = 0), S_0) \wedge \\ & PossiblyAfter([fwd, bk1], (pos = 1), S_0) \wedge \\ & CertainlyAfter([fwd, bk1, bk1], (pos = 0), S_0)\end{aligned}$$

Agent Actions Executability in NDsitCalc

The executability of an agent action sequence may depend on environment reactions.

Definitions

$CertainlyExecutable(\sigma, s)$ means that the sequence of agent actions σ is **certainly executable** no matter what the environment reactions are

$$CertainlyExecutable(\epsilon, s) \doteq True$$
$$CertainlyExecutable([A(\vec{x}), \sigma], s) \doteq$$
$$Poss_{ag}(A(\vec{x}), s) \wedge$$
$$\forall s'.Do_{ag}(A(\vec{x}), s, s') \supset CertainlyExecutable(\sigma, s')$$

$PossiblyExecutable(\sigma, s)$ means that the sequence of agent actions σ is **possibly executable**, i.e., executable for some environment reactions

$$PossiblyExecutable(\epsilon, s) \doteq True$$
$$PossiblyExecutable([A(\vec{x}), \sigma], s) \doteq$$
$$\exists s'.Do_{ag}(A(\vec{x}), s, s') \wedge PossiblyExecutable(\sigma, s')$$

Agent Actions Executability in NDsitCalc Track World Example

Example

For our running example $\mathcal{D}_{trk}$, we have:

$$\begin{aligned}\mathcal{D}_{trk} \models\ & CertainlyExecutable([fwd, bk1], S_0) \wedge \\ & PossiblyExecutable([fwd, bk1, bk1], S_0) \wedge \\ & \neg CertainlyExecutable([fwd, bk1, bk1], S_0)\end{aligned}$$

Regression in NDsitCalc

Theorem

$Do_{ag}(\sigma, s, s')$, $CertainlyAfter(\sigma, \phi, s)$, $PossiblyAfter(\sigma, \phi, s)$, $CertainlyExecutable(\sigma, s)$, and $PossiblyExecutable(\sigma, s)$ are all equivalent to sitCalc regressable formulas.

Outline

FOND Planning

- There has been much work on automated planning in **fully observable nondeterministic (FOND) domains**.
- In FOND domains, actions may have several alternative effects
- [DL21] shows how an arbitrary FOND domain (expressed in a FO PDDL) **can be translated into a NDBAT**.
- In a FOND domain, one may want to synthesize a **strong plan/strategy/policy that achieves a goal no matter how the environment behaves**
- Alternatively, one may want a strong cyclic plan that is guaranteed to achieve the goal under some fairness assumption

FOND Planning and Synthesis in NDsitCalc

Important concept: agent has a **strategy** that **forces a goal to become true** in spite of the environment reactions

Represent a **strategy** as a function from situation/history to agent action

Definition $AgtCanForceBy(Goal, s, f)$

i.e., **agent has a strategy** f that **forces** $Goal$ **to become true** in spite of the environment reactions:

$$AgtCanForceBy(Goal, s, f) \doteq \forall P.[\ \ldots\ \supset P(s)]$$

where $\ldots$ stands for

$$[(f(s) = stop \wedge Goal[s]) \supset P(s)] \wedge$$
$$[\exists A.\exists \vec{t}.(f(s) = A(\vec{t}) \neq stop \wedge Poss_{ag}(A(\vec{t}), s) \wedge$$
$$\forall e.(Poss(A(\vec{t}, e), s) \supset P(do(A(\vec{t}, e), s))))$$
$$\supset P(s)]$$

FOND Planning and Synthesis – Results

Theorem

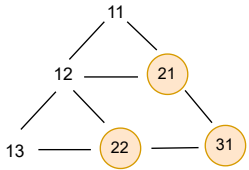
Let $Dom = (Flu, Const, Act, Init)$ be a FOND domain and $Goal$ a goal over it.

Let $\mathcal{D}$ be the NDBAT corresponding to Dom.

Then every strong plan f that achieves $Goal$ in Dom is a strategy such that $\mathcal{D} \models AgtCanForceBy(Goal, S_0, f)$ holds, and conversely any strategy f such that $\mathcal{D} \models AgtCanForceBy(Goal, S_0, f)$ holds is a strong plan for achieving $Goal$ in Dom.

Example: Triangle Tire World NDBAT $\mathcal{D}_l^{tt}$ [BDL23]

- Actions:
  - Agent Actions: $drive(o, d)$ and $fixFlatTire(l)$
  - System Actions: $drive(o, d, r)$, where $r = FlatTire \vee r = NoFlatTire$ and $fixFlatTire(l, r)$, where $r = Success$
- Fluents:
  - $At(l, s)$
  - $Flat(s)$
  - $Visited(l, s)$
  - ...
- Initial State Axioms:
  - $Road(o, d, S_0) \equiv (o, d) \in \{(11, 12), (11, 21), \ldots\}$
  - $Spare(l, S_0) \equiv l \in \{21, 22, 31\}$
  - $At(l, S_0) \equiv l = 11$, $Dest(l, S_0) \equiv l = 13$
  - $Visited(l, S_0) \equiv l = 11$



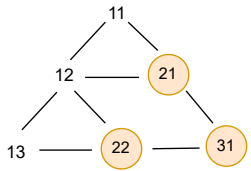
Example: Triangle Tire World NDBAT $\mathcal{D}_{tt}$

- Precondition Axioms:

$$
\begin{aligned}
&Poss_{ag}(drive(o,d),s) \doteq \\
&\quad o \neq d \wedge At(o,s) \wedge Road(o,d,s) \wedge \neg Flat(s) \\
&Poss(drive(o,d,r),s) \equiv \\
&\quad Poss_{ag}(drive(o,d),s) \wedge (r = FlatTire \vee r = NoFlatTire) \\
&Poss_{ag}(fixFlatTire(l),s) \doteq At(l,s) \wedge Spare(l,s) \wedge Flat(s) \\
&Poss(fixFlatTire(l,r),s) \equiv Poss_{ag}(fixFlatTire(l),s) \wedge r = Success
\end{aligned}
$$

- Successor State Axioms:

$$
\begin{aligned}
&At(l, do(a,s)) \equiv \exists o, r.a = drive(o,l,r) \vee \\
&\quad At(l,s) \ \wedge \forall d, r.a \neq drive(l,d,r) \\
&Flat(do(a,s)) \equiv \\
&\quad \exists o, d.a = drive(o,d,FlatTire) \vee \\
&\quad Flat(s) \wedge \forall r, l.a \neq fixFlatTire(l,r)
\end{aligned}
$$



FOND Planning in Triangle Tire World Example

E.g,, A strategy that guarantees reaching location 13

$$\mathcal{D}_l^{tt} \models AgtCanForceBy(At(13), S_0, f_l)$$

where

$$f_l(s) \doteq \begin{cases} stop & \text{if } At(13,s) \\ fixFlatTire(l) & \text{if } At(l,s) \wedge l \neq 13 \wedge Flat(s) \\ drive(o,d) & \text{if } At(o,s) \wedge o \neq 13 \wedge \neg Flat(s) \\ & \wedge\, Spare(d,s) \wedge Road(o,d,s) \\ & \wedge \neg Visited(d) \\ wait & \text{otherwise} \end{cases}$$

Outline

ConGolog Program Execution in Nondeterministic Domains

Can adapt the ConGolog semantics to apply to high-level programs involving **agent actions** executed in an NDBAT simply by taking:

$$Trans(A(\vec{x}), s, \delta', s') \equiv \exists e. Poss(A(\vec{x}, e), s) \wedge \delta' = \epsilon \wedge s' = do(A(\vec{x}, e), s)$$

But $Do(\delta, s, s')$ just means that situation s' can be reached by executing program δ in s if both the agent and environment cooperate.

We define $AgtCanForceBy(\delta, s, f)$, i.e., strategy f **successfully executes** the nondeterministic program δ considering its nondeterminism angelic, but also **considering the nondeterminism of environment reactions devilish** (i.e., adversarial).

ConGolog Program Execution in Nondeterministic Domains

Definition $AgtCanForceBy(\delta, s, f)$

i.e., agent can **successfully execute** the nondeterministic program δ using strategy f in spite of environment reactions:

$$AgtCanForceBy(\delta, s, f) \doteq \forall P.[\ \ldots\ \supset P(\delta, s)] \wedge$$

where ... stands for

$$[(f(s) = stop \wedge \mathit{Final}(\delta, s)) \supset P(\delta, s)] \wedge$$

$$[\exists A.\exists \vec{t}.(f(s) = A(\vec{t}) \neq stop \wedge$$
$$\exists e.\exists \delta'.\mathit{Trans}(\delta, s, \delta', do(A(\vec{t}, e), s)) \wedge$$
$$\forall e.(\exists \delta'.\mathit{Trans}(\delta, s, \delta', do(A(\vec{t}, e), s)) \supset$$
$$\exists \delta'.\mathit{Trans}(\delta, s, \delta', do(A(\vec{t}, e), s)) \wedge P(\delta', do(A(\vec{t}, e), s))))$$
$$\supset P(\delta, s)]$$

For programs that are not situation determined and may have more than one remaining program after a given action, we can also introduce a second strategy function to select the remaining program

Strategic Ability to Execute Program in Track World Example

For running e.g., we have;

$$\mathcal{D}_{trk} \models AgtCanForceBy(\delta_2, S_0, f)$$

where $\delta_2 \doteq (fwd|bk1)^*; pos(s) = Dest?$

and $f(s) = \begin{cases} fwd & \text{if } pos(s) < Dest, \\ bk1 & \text{if } pos(s) > Dest, \\ stop & \text{if } pos(s) = Dest \end{cases}$

Outline

Conclusion

NDsitCalc Features

- We have seen a **nondeterministic variant of the situation calculus**
- It separates **agent actuation of the action** from the **environment reaction**, which determines the action's outcome
- We retain a notion of **system action** formed by the agent's action together with the environment's reaction, as in the standard SitCalc
- But we can **quantify separately** over agent actions and environment reactions
- Preserves Reiter's solution to the frame problem and answering projection queries through regression
- We have also seen how to handle FOND planning and ConGolog high-level program execution in nondeterministic domains

Conclusion

NDsitCalc Features

- The predicates that define planning in nondeterministic domains and nondeterministic program execution, must be expressed in second order logic and are essentially fixpoint formulas
- Makes reasoning challenging in general
- But in the finite domain/propositional case can be reduced to Mu-Calculus formulas over situations, or configurations formed by pairs of programs and situations
- So checking these properties and synthesizing strategies/plans is **decidable**
- Remains decidable for NDBATs with infinitely many object that are state bounded

References

[DL21] Giuseppe De Giacomo, Yves Lespérance: The Nondeterministic Situation Calculus. KR 2021:, 216-226.

[BDL23] Bita Banihashemi, Giuseppe De Giacomo, Yves Lespérance: Abstraction of Nondeterministic Situation Calculus Action Theories. IJCAI 2023: 3112-3122