Automatic Synthesis of New Behaviors from a Library of Available Behaviors

Giuseppe De Giacomo

Università di Roma "La Sapienza", Roma, Italy

degiacomo@dis.uniroma1.it

Sebastian Sardina

RMIT University, Melbourne, Australia

ssardina@cs.rmit.edu.au

Behavior composition

Target behavior
description of the desired behavior
expressed in terms of virtual actions

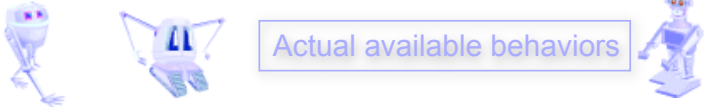
Environment *is similar to an action theory!*

Environment
description of (virtual) actions, preconditions and effects

Behaviors *are similar to robot programs; capture possible executions*

Available behaviors
descriptions of the behavior of available agents/devices
expressed in terms virtual actions

. . .



Actual available behaviors

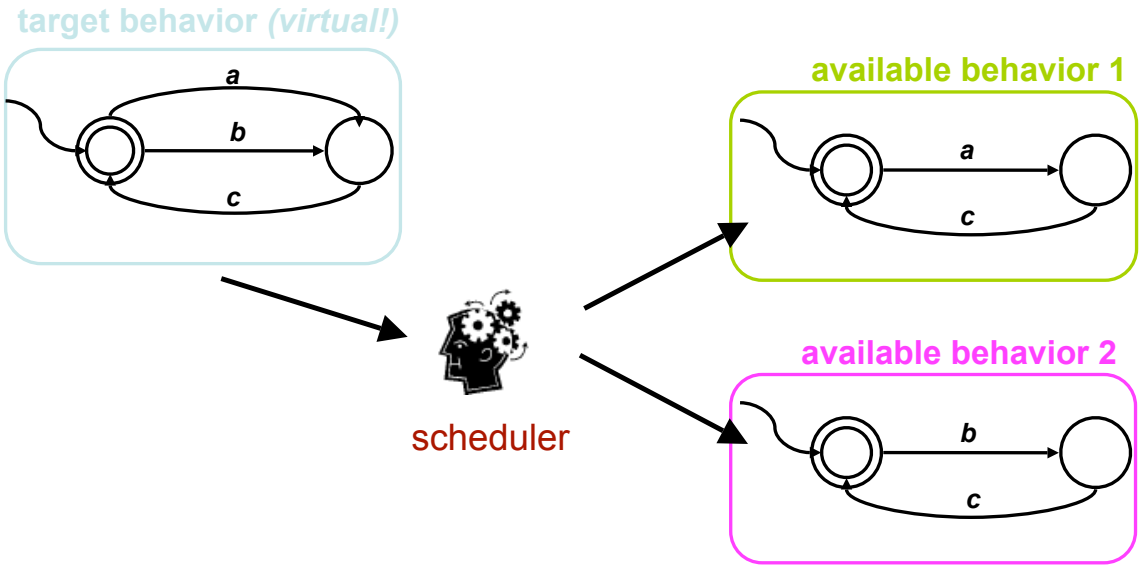
Key points

- ***Actions*** *are* ***virtual***
- *Only* ***available behaviors*** *provide* ***actual action execution***
- *Must realize* ***target behavior*** *using* ***fragments*** *of* ***available behaviors***

Behavior composition:
the setting studied

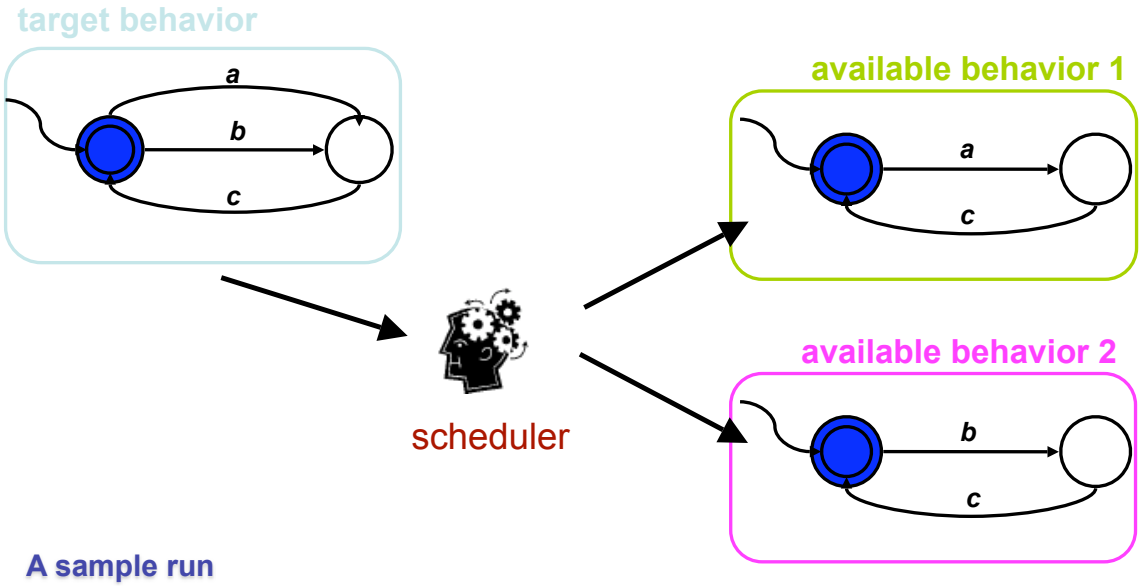
- **Environment:**
  - *Describe precondition and effect of actions (as an action theory)*
  - **Finite state** (to get computabiliy of the synthesis)
  - **Nondeterministic** (devilish/don't know nondeterminism)
  - Represented as a (finite) **transition system** (we are not concerned with representation in this work)
- **Available behaviors:**
  - *Describe the capabilities of the agent/device*
  - **Finite state** (to get computability of the synthesis)
  - **Nondeterministic** (devilish/don't know nondeterminism)
  - Can **access** the state of the **environment**
  - Can **not access** the state of the **other available behaviors**
  - Represented as (finite) **transition systems** (with **guards** to test the environment)
- **Target behavior:**
  - As available behavior but **deterministic**
    - it's a spec of a desired behavior: we know what we want!
- *Problem: **synthesize a "scheduler"** that realize the target behavior by suitably "composing" the available behaviors*

Example



Simplified case: available behaviors are deterministic finite transition systems

Example

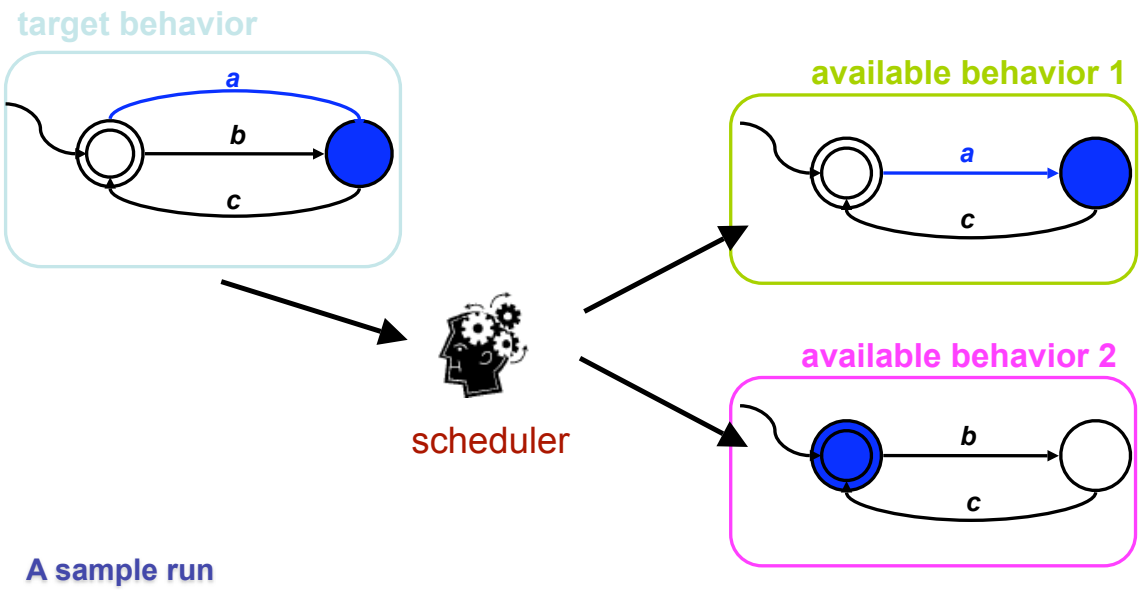


A sample run

action request:

scheduler response:

Example

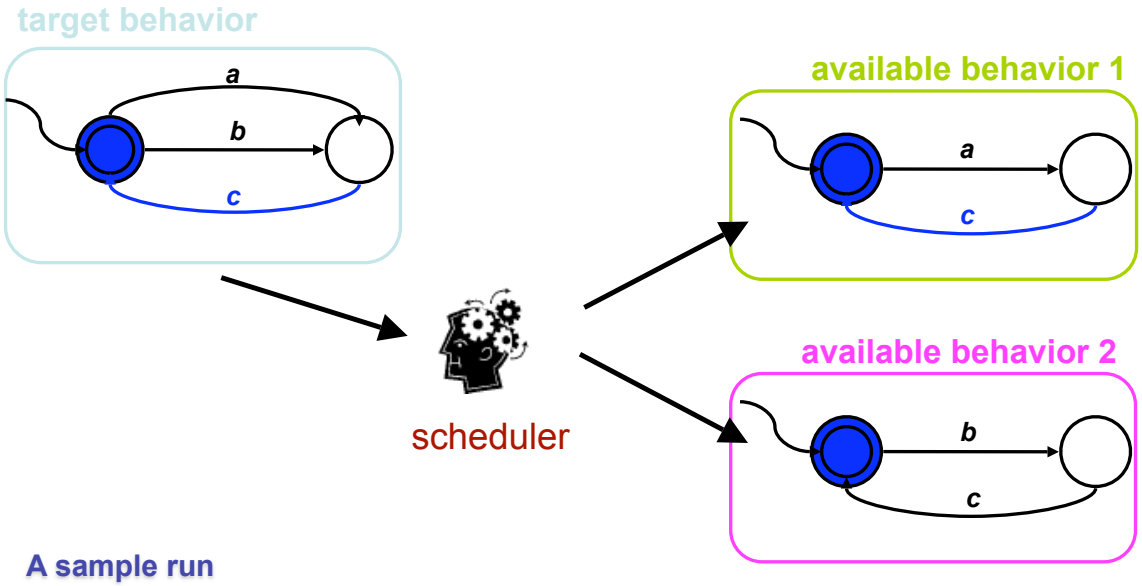


A sample run

action request: *a*

scheduler response: *a,1*

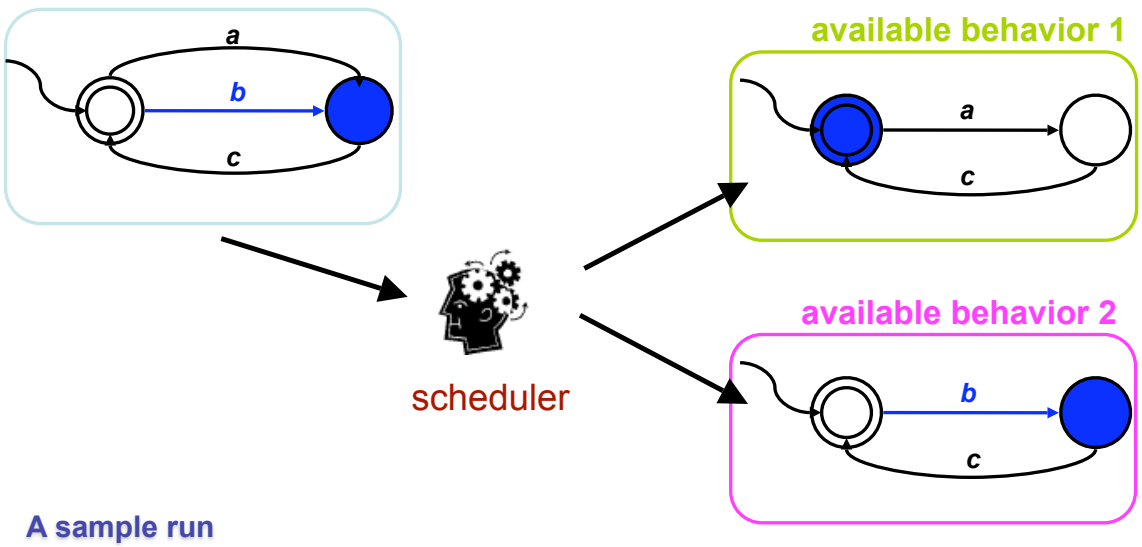
Example



A sample run

action request:	*a*	*c*
scheduler response:	*a,1*	*c,1*

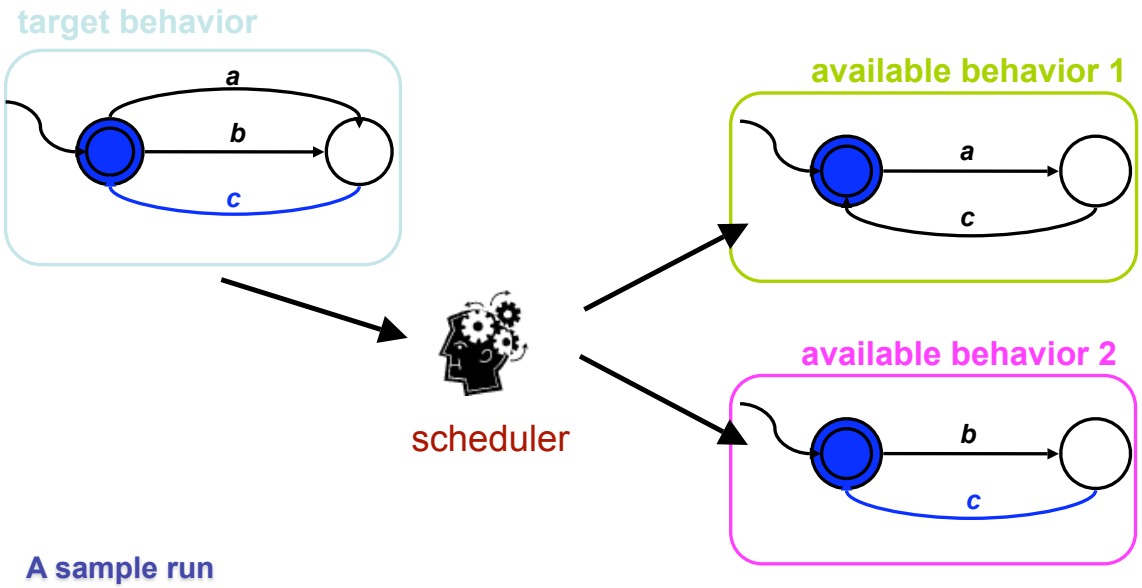
Example



A sample run

action request:	*a*	*c*	*b*
scheduler response:	*a,1*	*c,1*	*b,2*

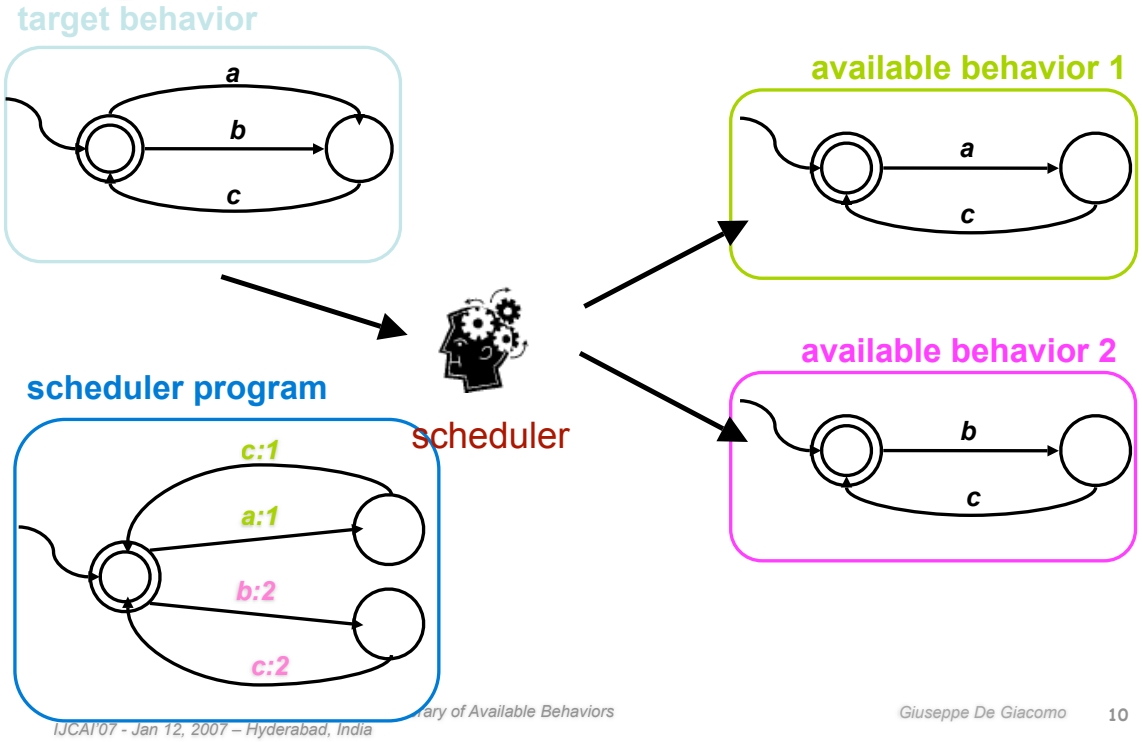
Example



A sample run

action request:	*a*	*c*	*b*	*c*	…
scheduler response:	*a,1*	*c,1*	*b,2*	*c,2*	

A scheduler program realizing the target behavior



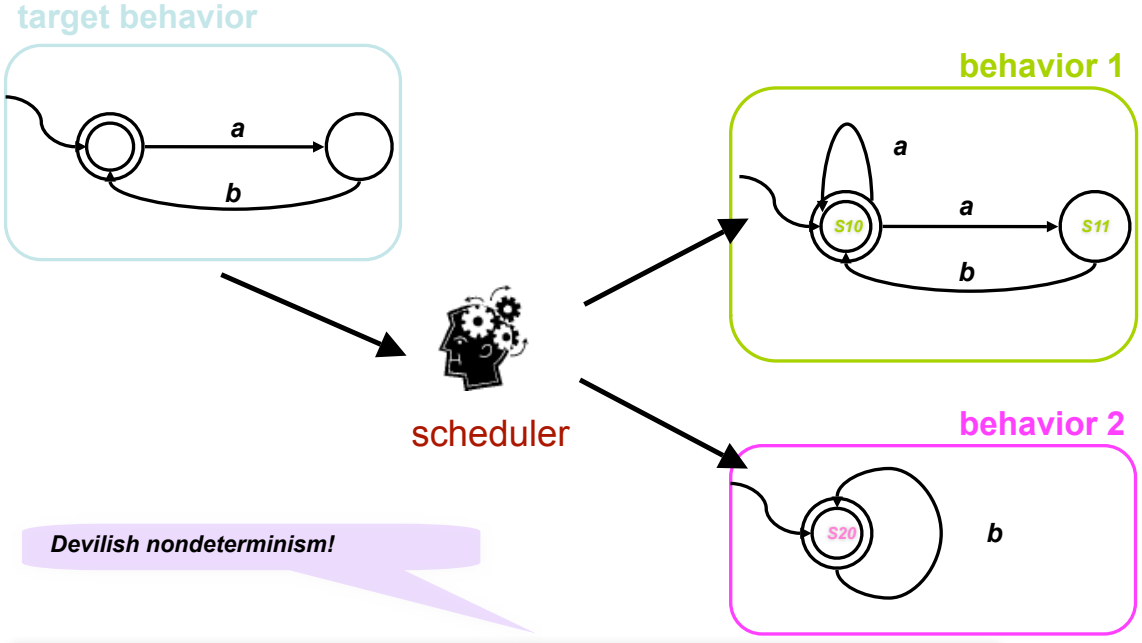
Nondeterminism

Devilish (don't know)!

- Nondeterministic environment
  - Incomplete information on effects of actions
  - Action outcome depends on external (not modeled) events
- Nondeterministic available behaviors
  - Incomplete information on the actual behavior
  - Mismatch between behavior description (which is in terms of the environment actions) and actual behavior of the agents/devices
- Deterministic target behavior
  - it's a spec of a desired behavior: (devilish) nondeterminism is banned

In general, devilish nondeterminism difficult to cope with
eg. nondeterminism moves AI Planning from PSPACE (classical planning) to EXPTIME (contingent planning with full observability [Rintanen04])

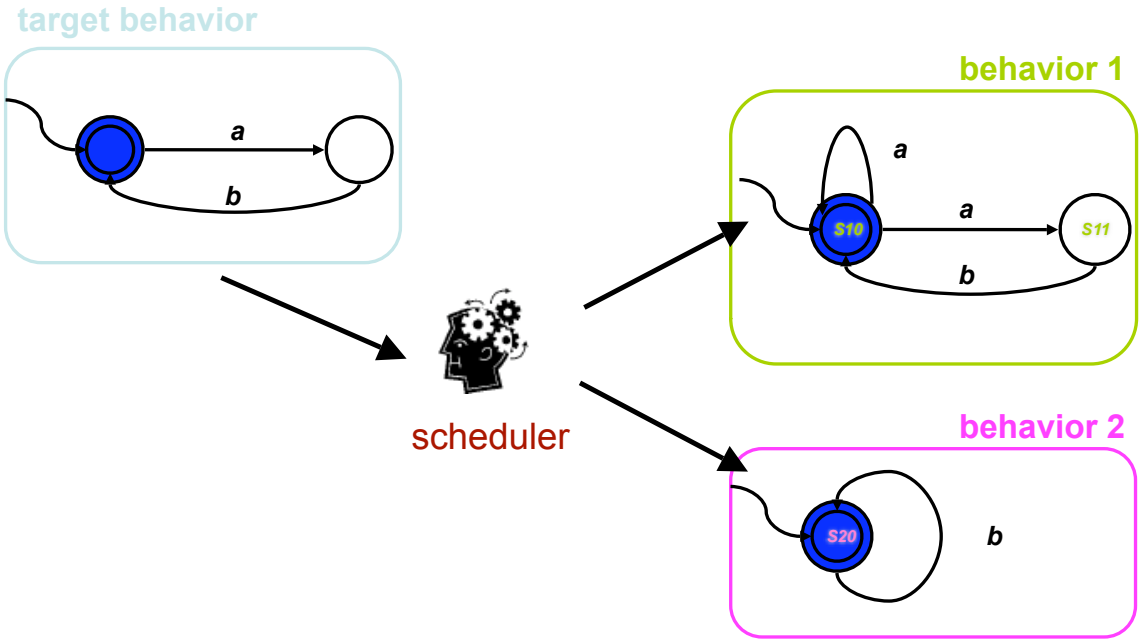
Example nondeterministic behaviors



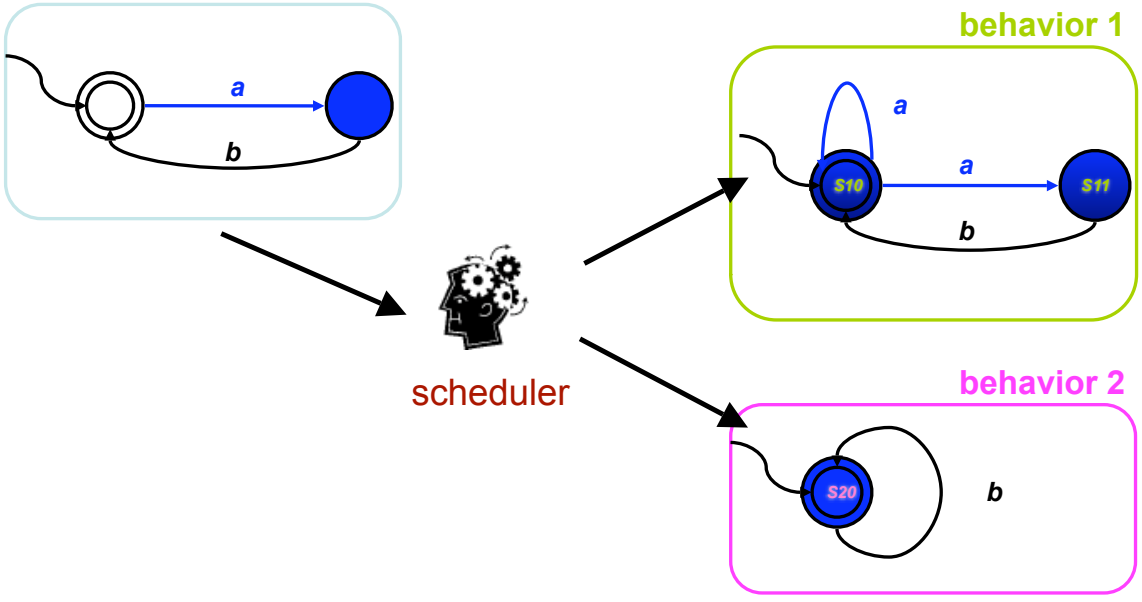
Devilish nondeterminism!

Available behaviors represented as nondeterministic transition systems

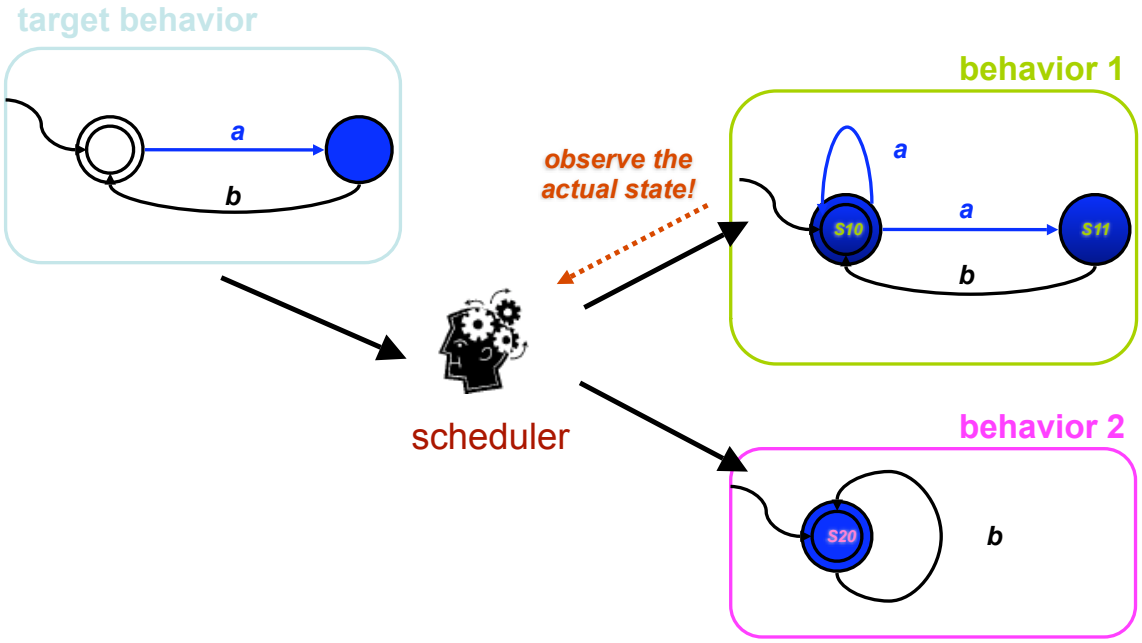
Example nondeterministic behaviors



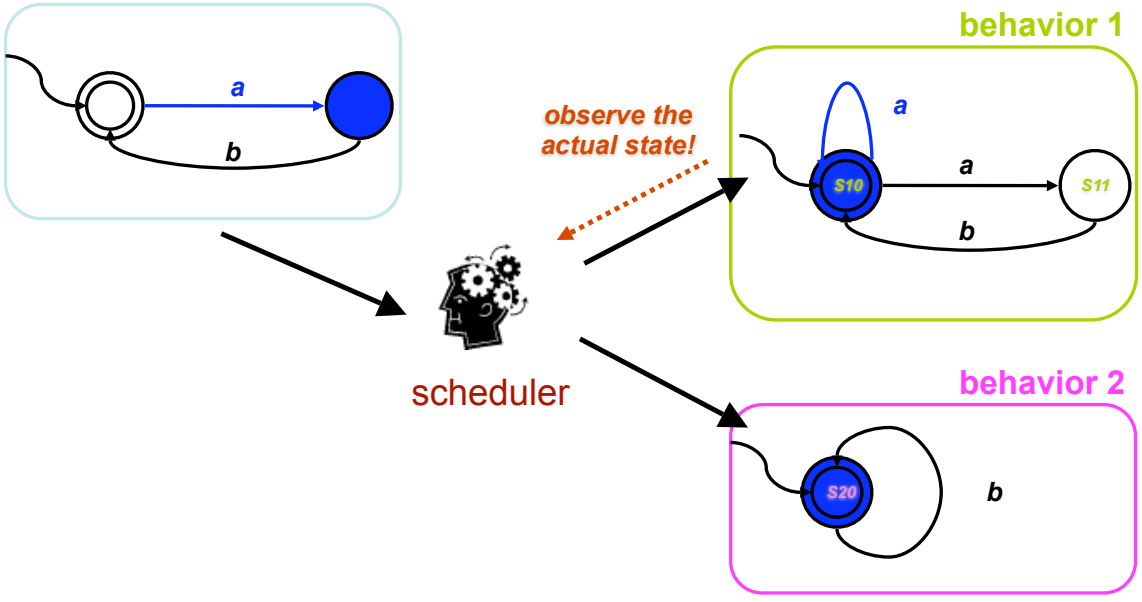
Example nondeterministic behaviors



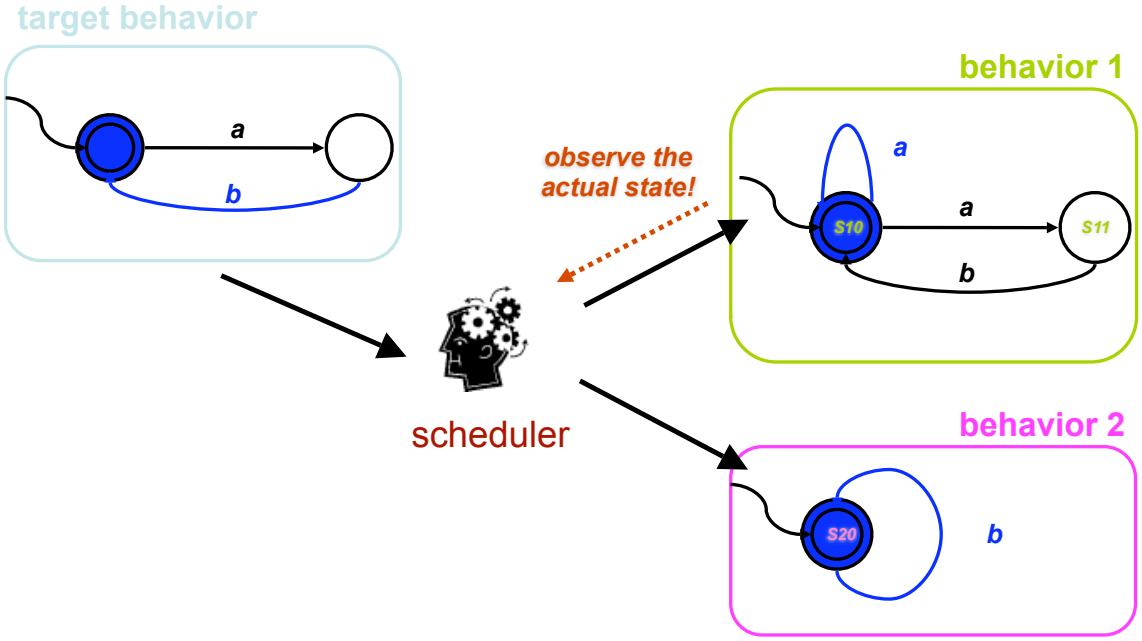
Example nondeterministic behaviors



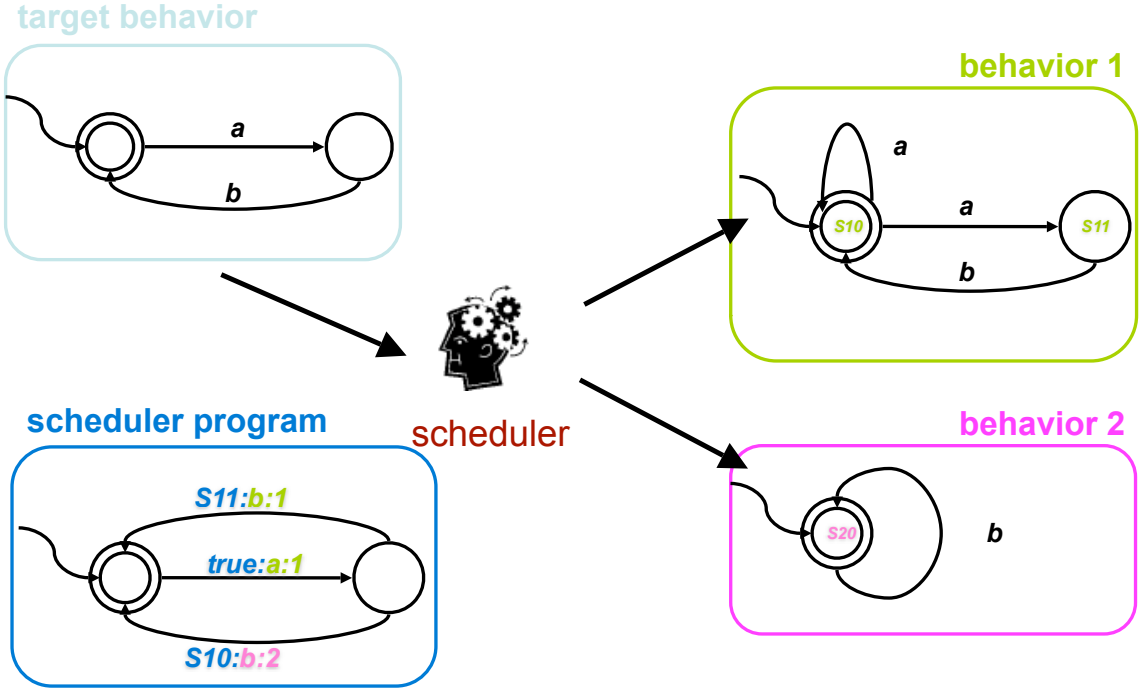
Example nondeterministic behaviors



Example: nondeterministic behaviors



A scheduler program realizing the target behavior



Scheduler programs

contains all the observable information up the current situation

- **Scheduler program** is *any function P(h,a) = i* that takes a **history** *h* and an **action** *a* to execute and **delgates** *a* to the available behavior *i*
- A **history** is a sequence of the form:

  $(s_1^0,s_2^0,\ldots,s_n^0,e^0)\ a_1\ (s_1^1,s_2^1,\ldots,s_n^1,e^1)\ \ldots\ a_k\ (s_k^1,s_2^k,\ldots,s_n^k,e^k)$

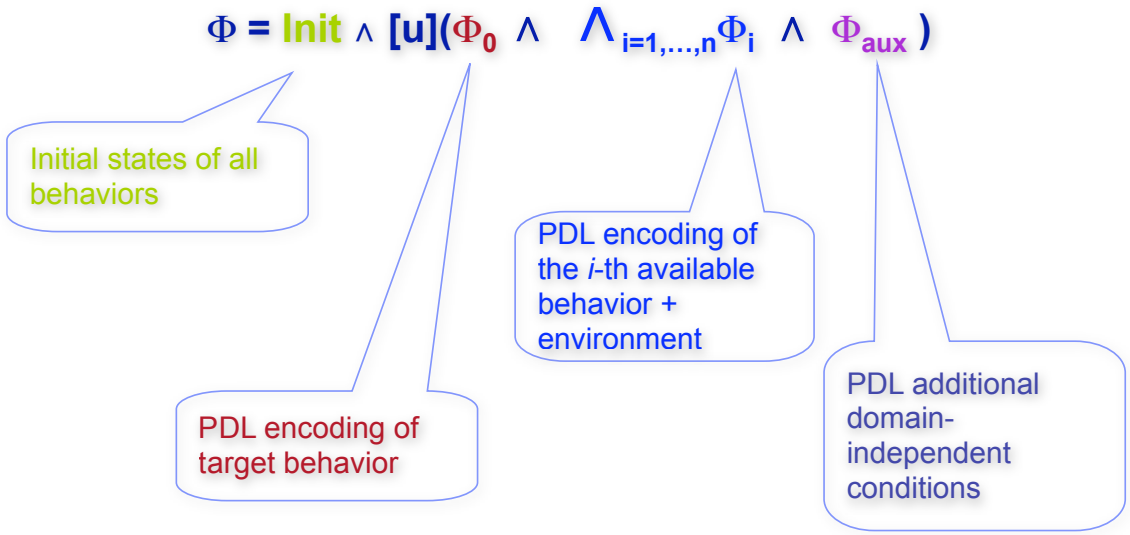
- Observe that to take a decision *P* has **full access to the past**, but no access to the future
- *Problem: synthesize a scheduler program P that realizes the target behavior making use of the available behaviors*

Technique: reduction to PDL

Basic idea:

- A scheduler program *P* realizes the target behavior *T* iff:
  - ∀ transition labeled *a* of the target behavior *T* …
  - … ∃ an available behavior B_i (the one chosen by *P*) which can make an *a*-transition …
  - … and ∀ *a*-transition of B_i realizes the *a*-transition of *T*
- Encoding in PDL:
  - ∀ transition labeled *a* …
    use **branching**
  - ∃ an available behavior B_i …
    use underspecified predicates **assigned through SAT**
  - ∀ *a*-transition of B_i … :
    use **branching** again

Structure of the PDL encoding



PDL encoding is polynomial in the size of the target behavior, available behaviors, and environment

Technical results: theoretical

Thm Checking the existence of scheduler program realizing the target behavior is **EXPTIME-complete**.

EXPTIME-hardness due to Muscholl&Walukiewicz05 for deterministic behaviors

Thm If a **scheduler program exists** there exists one that is **finite state**.

Exploits the finite model property of PDL

Technical results: practical

Reduction to PDL provides also a practical sound and complete technique to compute the scheduler program

eg, PELLET @ Univ. Maryland

- Use state-of-the-art tableaux systems for OWL-DL for checking SAT of PDL formula F

exponential in the size of the behaviors

- *If SAT, the tableau returns a finite model of F*
- Project away irrelevant predicates from such model, and possibly minimize

polynomial in the size of the model

- *The resulting structure is a finite scheduler program that realizes the target behavior*

Conclusion

- Nondeterministic target behavior?
  - loose specification in client request
  - **angelic (don't care)** vs devilish (don't know) nondeterminism
  - see ICSOC'04 for ideas
- **Distribute the scheduler**?
  - Often a centralized scheduler is unrealistic: eg. Robot Ecologies
    - too tight coordination
    - too much communication
    - scheduler cannot be embodied anywhere
  - drop centralized scheduler in favor of **independent controllers** on single available behaviors (exchanging messages)
  - we are actively working on it
- Infinite states behaviors?
  - Important for dealing with **data**/parameters
  - this is the single most difficult issue to tackle
    - first results: actions as DB updates, see VLDB'05
    - literature on Abstraction in Verification

PDL encoding: target behavior

For target behavior B_0 : Φ_0 is the conjunction of

- $s \wedge e \rightarrow \langle a \rangle \text{true} \wedge [a]s'$ for each $(s,g,a,s') \in \delta_0$ with $g(e) = \text{true}$
  target behavior can do an a-transition going to state s'
- $s \wedge e \rightarrow [a]\ \text{undef}$ if there exists no $(s,g,a,s') \in \delta_0$ with $g(e) = \text{true}$
  target behavior does not do an a-transition
- $s \rightarrow \neg s'$ for all pairs of distinct states of the behaviors
  behavior states are pair-wise disjoint
- $F_o \equiv \bigvee_{s \in Fo} s$
  denotes behavior final states

PDL encoding: available behaviors

For available behavior B_i: Φ_i is the conjunction of

- $s \wedge e \wedge \text{EXEC}_{ia} \rightarrow \bigwedge_{(s',e') \in \Delta} \langle a \rangle (s' \wedge e') \wedge [a]_{(s',e') \in \Delta} (s' \wedge e')$
  where $\Delta = \{(s' \wedge e') \mid (s,g,a,s') \in \delta_i, g(e) = \text{true}\}$ ((*If behavior is select to be executed (EXEC_i is true) then for each action that has nonempty transitions, it moves in all possible way*
- $s \wedge e \wedge \text{EXEC}_{ia} \rightarrow [a]\ \text{false}$ if $\Delta = \emptyset$
  if behavior is selected to be executed and it cannot do a, then there is no a-transition
- $s \wedge \neg \text{EXEC}_i \rightarrow [a]s$
  if behavior is not selected to be executed then it remains in its state
- $s \rightarrow \neg s'$ for all pairs of distinct states of the behaviors
  behavior states are pair-wise disjoint
- $F_i \equiv \bigvee_{s \in Fi} s$
  denotes behavior final states

PDL encoding: additional conditions

Finally: Φ_{aux} is the conjunction of

- $\text{undef} \rightarrow [a]\,\text{undef}$

  successors of undef states are undef themselves

- $\neg\,\text{undef} \wedge \langle a\rangle\text{true} \rightarrow \bigvee_{1,\ldots,n} \text{EXEC}_{ia}$

  at least one of the available behaviors must be selected for execution at each step

- $\text{EXEC}_{ia} \rightarrow \neg\,\text{EXEC}_{ja}$

  only one available behavior can execute at each step

- $F_0 \rightarrow \bigwedge_{i=1,\ldots,n} F_i$

  when target behavior is final all available behaviors are final

and **Init** is

- $\text{Init} \equiv s^0_0 \wedge_{i=1\ldots n} s^0_i$