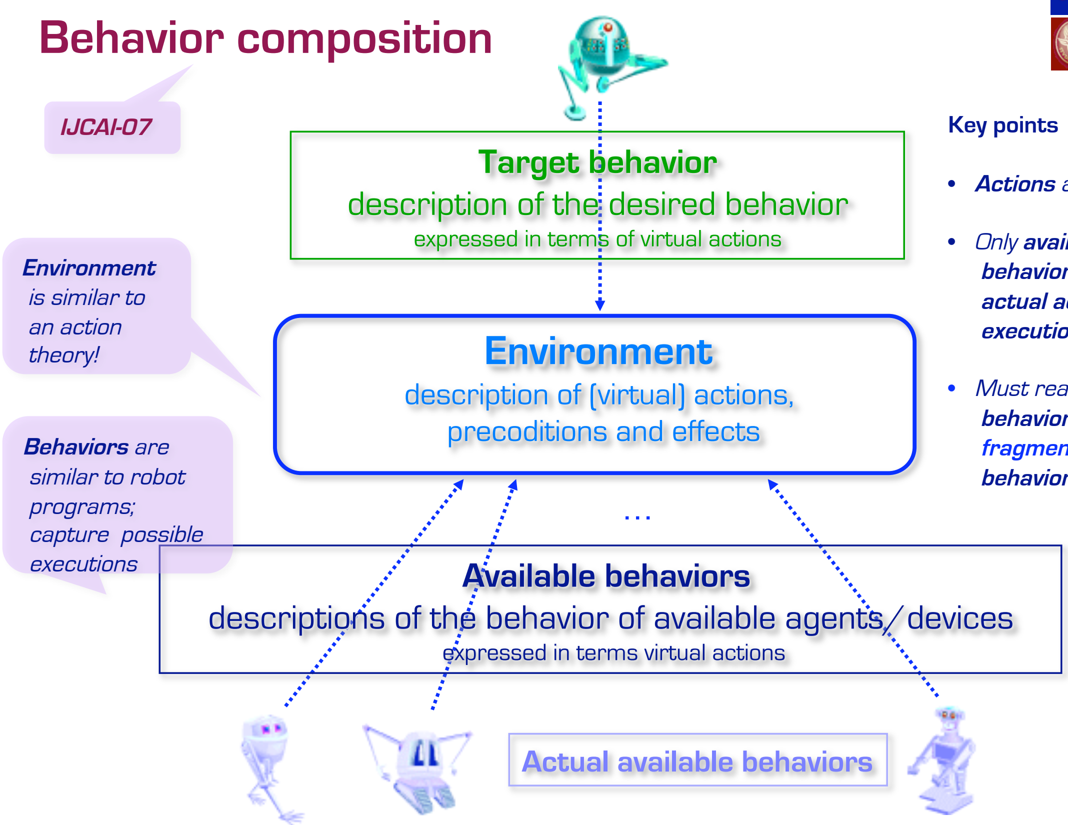
Automatic Synthesis of a Global Behavior from Multiple Distributed Behaviors

Sebastian Sardina
RMIT University, Melbourne, Australia
ssardina@cs.rmit.edu.au

Fabio Patrizi & Giuseppe De Giacomo
Sapienza Università di Roma, Italy
lastname.uniroma1.it

Behavior composition



Key points

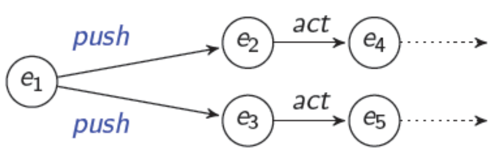
- ***Actions*** are ***virtual***
- *Only* ***available behaviors*** *provide* ***actual action execution***
- *Must realize* ***target behavior*** *using* ***fragments*** *of* ***available behaviors***

Behavior composition: setting studied

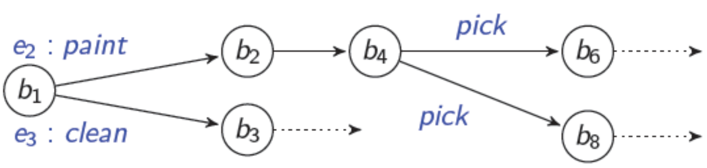
- **Environment:**
  - *Describe precondition and effect of actions (as an action theory)*
  - **Finite state** (to get computabiliy of the synthesis)
  - **Nondeterministic** (devilish/don't know nondeterminism)
  - Represented as a (finite) **transition system** (we are not concerned with representation 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!

Behavior composition: setting studied

- Nondeterministic environment:
  - Incomplete information on effects of actions in the domain
  - Action outcome depends on external (not modeled) events

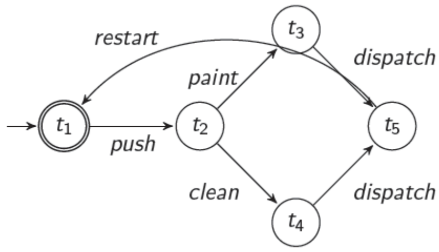


- Nondeterministic available behaviors:
  - Incomplete information on the actual behavior
  - Mismatch between behavior description and actual agents/devices



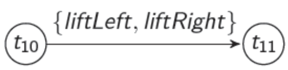
Behavior composition: setting studied

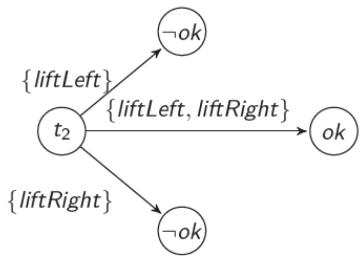
- Deterministic target behavior:
  - It's a spec of the desired global behavior



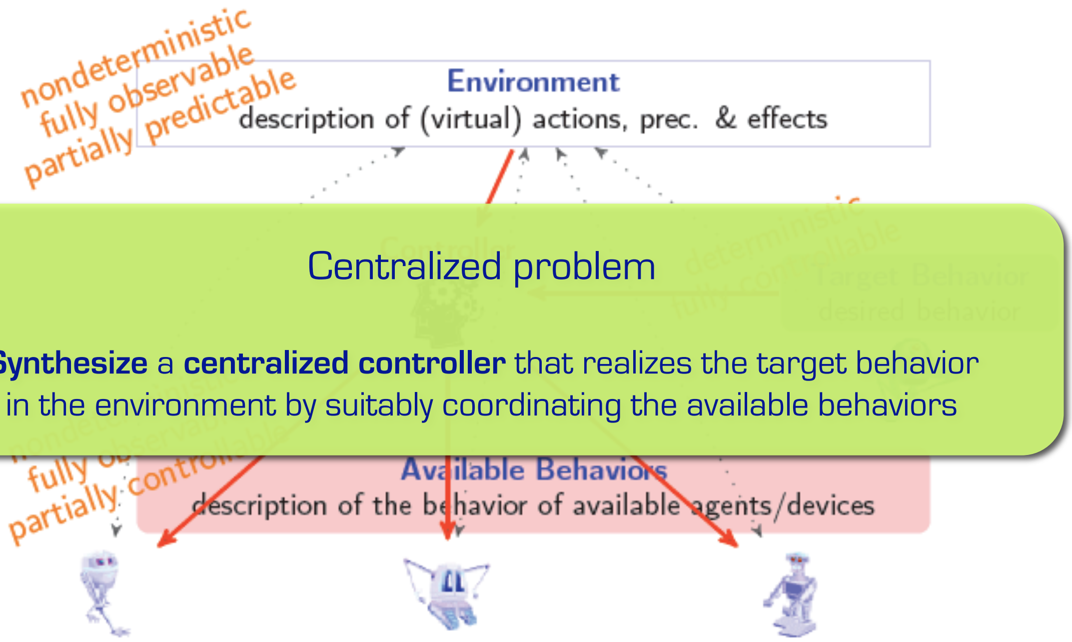
- Concurrent actions allowed (in preparation for distribution):
  - multiple actions performed at the same time.
  - environment, available behaviors, target behavior allow for concurrent actions

New for AAAI'07





Behavior composition: centralized version

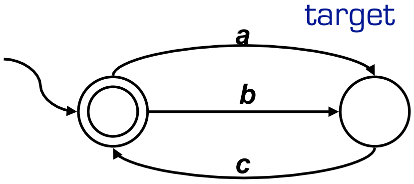


Centralized problem

Synthesize a **centralized controller** that realizes the target behavior in the environment by suitably coordinating the available behaviors

Example: centralized controller

target

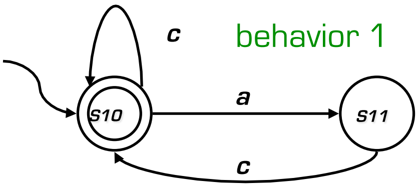


controller

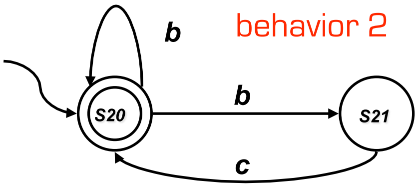
NextAct:
act = a
delegate 1
act = b
delegate 2
act = c
if (state2 = S21) delegate 2
else delegate 1

NextState: -- //it's stateless

behavior 1

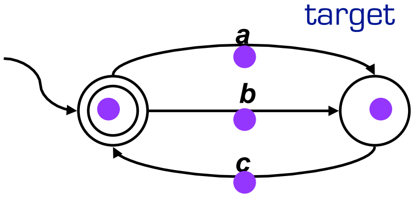


behavior 2



Example: centralized controller

target



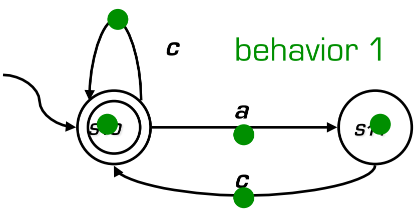
acbcbc

controller

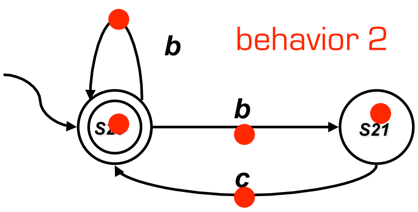
NextAct:
act = a
delegate 1
act = b
delegate 2
act = c
if (state2 = S21) delegate 2
else delegate 1

NextState: -- //it's stateless

behavior 1



behavior 2



Contributions on centralized problem

- Technique for automatic synthesis of the centralized controller based on PDL satisfiability (deals with concurrent actions!)

- Computational complexity characterization of the problem: EXPTIME-complete (deals with concurrent actions!)

- Normal form for the controller:
  - nexta: $\Sigma \times S_1 \times \ldots \times S_n \times E \times 2^A \rightarrow 2^A$
  - nexts: $\Sigma \times E \times 2^A \rightarrow \Sigma$

Finite state!!!

Independent from available service states

Bounded by #states of target service

bound is tight

Distribute the controller?

- Composition problem is of particular interest in settings where components are distributed and no central mediator is available.

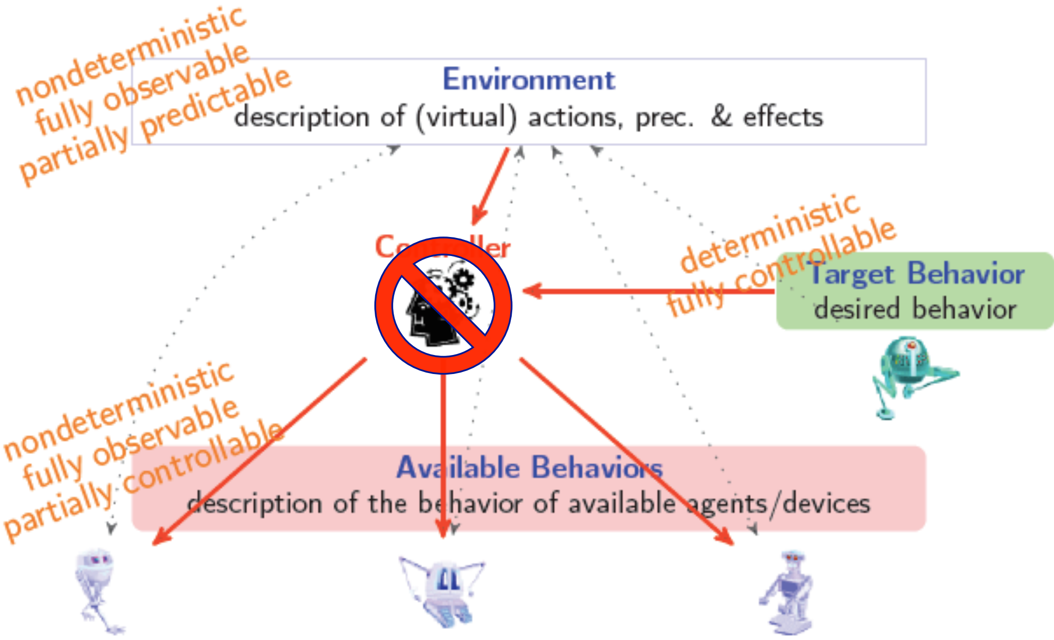
- Examples: independent RoboCup players; robot ecologies; peer devices.

- In such cases, behavior components are not controllable as a whole.

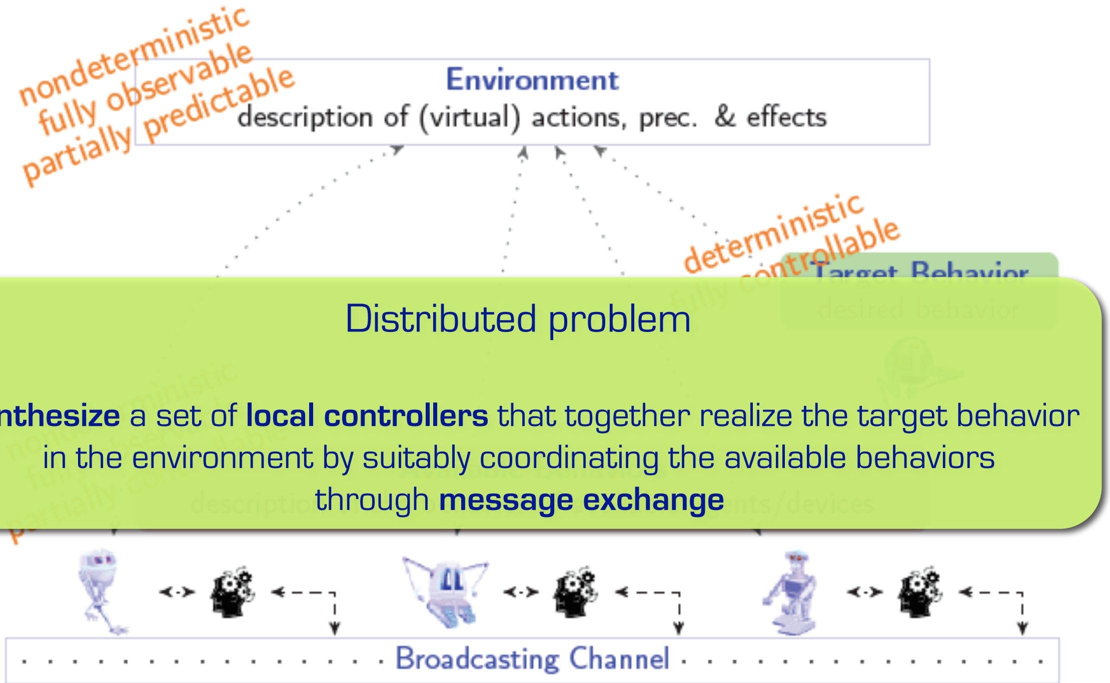
- Thus, unrealistic to rely on a centralized controller!
  - too tight coordination
  - too much communication
  - controller cannot be embodied anywhere

- Nonetheless, we can rely on:
  - local control of each component (via "local controllers");
  - some kind of communication among such local controllers.

Behavior composition: distributed version



Behavior composition: distributed version



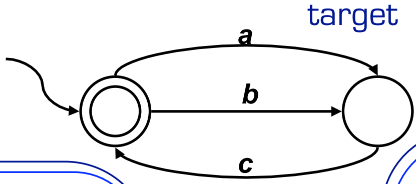
Distributed problem

Synthesize a set of **local controllers** that together realize the target behavior in the environment by suitably coordinating the available behaviors through **message exchange**

Example: distributed controller

RMIT University
SAPIENZA Università di Roma

target

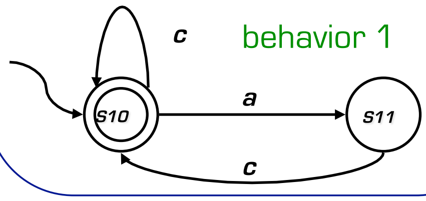


controller 1

NextAct:
act = a
do a
act = b
noop
act = c
if ((S10 and ”2: cannot do c”) or S11)
do c
else noop

NextMsg: -- //does not generate msgs
NextState: -- //it’s stateless

behavior 1



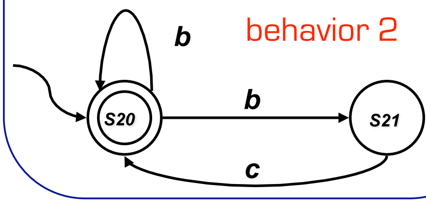
controller 2

NextAct:
act = a
noop
act = b
do b
act = c
if ((S21) do c
else noop

NextMsg:
if (act = c and S20)
send "2:cannot do c"

NextState: -- //it’s stateless

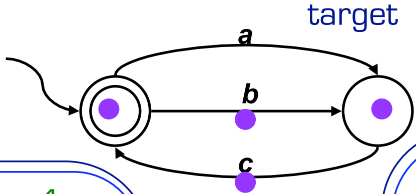
behavior 2



Example: distributed controller

RMIT University
SAPIENZA Università di Roma

target



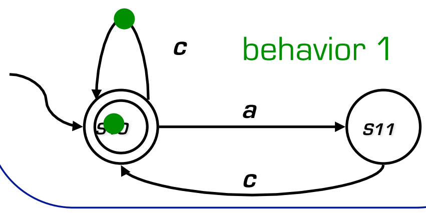
bcbc

controller 1

NextAct:
act = a
do a
act = b
noop
act = c
if ((S10 and ”2: cannot do c”) or S11)
do c
else noop

NextMsg: -- //does not generate msgs
NextState: -- //it’s stateless

behavior 1



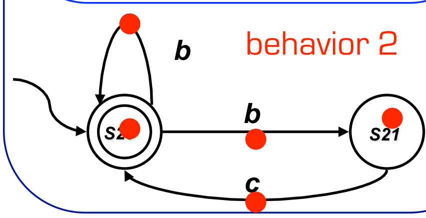
controller 2

NextAct:
act = a
noop
act = b
do b
act = c
if ((S21) do c
else noop

NextMsg:
if (S20 after b)
send ”2: cannot do c"

NextState: -- //it’s stateless

behavior 2



Distributed problem vs centralized problem: how they compare?

- *Does the distributed problem admit solutions when the centralized does not? Or is the converse true?*
- *Is the distributed problem more/less complex from the computational point of view?*
- *Are there techniques for the distributed problem?*
- We study these issues under the following assumptions:
  - Local controllers can synchronize on actions issued by the target
  - Local controllers can exchange an unlimited amount of message types, but once per action
  - Channel is fully reliable

"Pure setting": no further constraints except for the distribution itself

Contribution on distributed problem

- Thm: distributed problem as a solution iff the centralized problem does.

  ⇒ Thus, the computational complexity of the two problems is identical (EXPTIME-complete)
- Thm: from a finite central controller one can obtain the set of finite local controllers (exchanging finite message types) for the distributed problem. And vice-versa.
- Thm: message types required are bounded by #states of the available behaviors

bound is tight

- ⇒ We have a technique to automatically synthesize finite local controllers exchanging message types (bounded as above).

Local controllers from central ones: how to

- Finite local controllers have the form:
  - $nexta_i: \Sigma \times S_i \times E \times 2^M \times 2^A \to 2^A$
  - $nextm_i: \Sigma \times S_i \times E \times 2^M \times 2^A \times S_i \to 2^M$
  - $nexts_i: \Sigma \times S_i \times E \times 2^M \times 2^A \to \Sigma$

- Form centralized controller in normal form define local controllers:
  - $nexta: \Sigma \times S_1 \times \ldots \times S_n \times E \times 2^A \to 2^A$
  - $nexts: \Sigma \times E \times 2^A \to \Sigma$

  ... define local controller as:
  - $nexta_i[\sigma,s_i,e,\{``1{:}s_1",\ldots,``n{:}s_n"\},A] = A_i$ iff $nexta[\sigma,s_1,\ldots,s_n,e,A] = [A_1,\ldots,A_i,\ldots,A_n]$;
  - $nextm_i[\sigma,s_i,e,,\{``1{:}s_1",\ldots,``n{:}s_n"\},A,s'] = ``i{:}s'_i"$ iff $s'_i \in \delta[s_i, nexta_i[\sigma,s_i,e,\{``1{:}s_1",\ldots,``n{:}s_n"\},A]\,]$
  - $nexts_i[\sigma,\text{-},e,\text{-},A] = nexts[\sigma,e,A]$.

Optimal!!!

A_i

Look up at current state of behavior is not need

Look up at current messages is not needed

Conclusions

- Further work
  - More efficient synthesis techniques based on the formal notion of simulation
    - to take advantages of symbolic techniques at the base of model checking
  - Robust solutions:
    - use simulation for with just-in-time composition
  - Multiple target behaviors:
    - build virtual agents community instead of virtual isolated agents
  - Partial observable environment and available behavior states
    - not so simple extension
- Specific further work on distributed problem
  - More asynchronous accounts
    - to avoid synchronizing at each action
  - Consider limits of the communication channel
    - a priori limited bandwidth: a form of further constraints on the solution
    - unreliable channel: robust solutions
  - Point-to-point communication instead of broadcasting
    - a form of further constraints on the solution
  - Local environments instead of a single shared environment
    - simple extension, but interesting in practice