Petri Net Plans

A Formal Model for Representation and Execution of Plans

V. A. Ziparo, L. Iocchi, D. Nardi

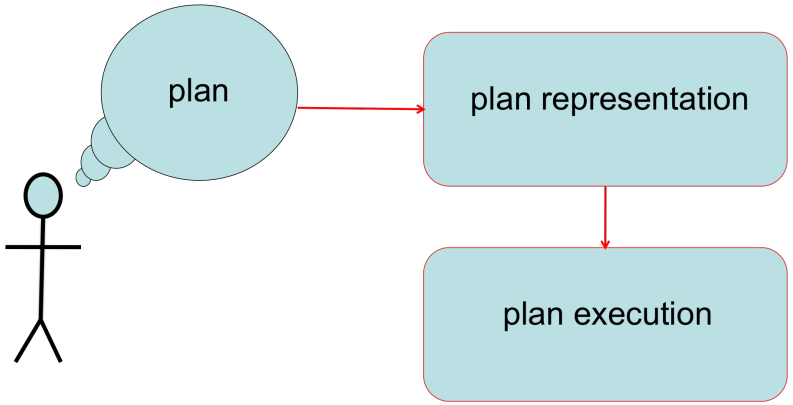
RoCoCo Laboratory
Dipartimento di Ingegneria Informatica Automatica e Gestionale “Antonio Ruberti”
“Sapienza” Università di Roma

Outline

Introduction

Petri Net Plans

A formal model for Distributed Execution of multi-robot plans based on Petri nets.



Plans for Robots are though!

Complex Behaviors with Hard Real Time Constraints

- Information Gathering during Execution
- Non-Instantaneous Actions, Concurrency and Failures
- Multi-Robot specification and Distributed Execution

Suitable for Analysis

- Reachability of goal states
- Deadlock free
- ...

Intuitive

- Easy to Understand
- Easy to Debug

Related Work

High Level Robot Programming

1. Hand-written behaviors *directly coded in the robot program*.
   - No explicit representation of actions and plans.
2. Hand-written behaviors using *behavior oriented languages* (e.g. Xabsl [LBBJ04] and Colbert [Kon97]).
   - User must code behavioral routines.
3. Hand-written behaviors using formal models (i.e., Petri Nets)
   - PN used but for single-robot [CL07] (no robot implementation yet)
   - PN specific coordination tasks [SY05] (no general framework)

Outline

Petri Net Definition

Definition

$PN = \langle P, T, F, W, M_0 \rangle$

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of *places*.
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of *transitions*.
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of edges.
- $W : F \rightarrow \{1, 2, 3, \ldots\}$ is a weight function and $w(n_s, n_d)$ denotes the weight of the edge from n_s to n_d.
- $M_0 : P \rightarrow \{0, 1, 2, 3, \ldots\}$ is the initial marking.
- $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$

Graphical Representation

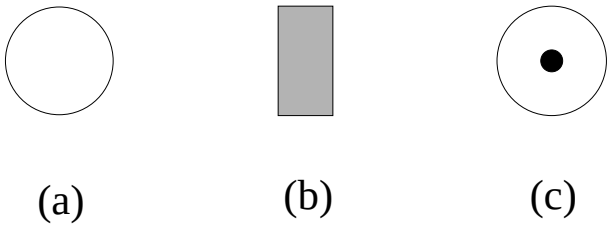


Figure: (a) A place. (b) A Transition. (c) A Place with one token.

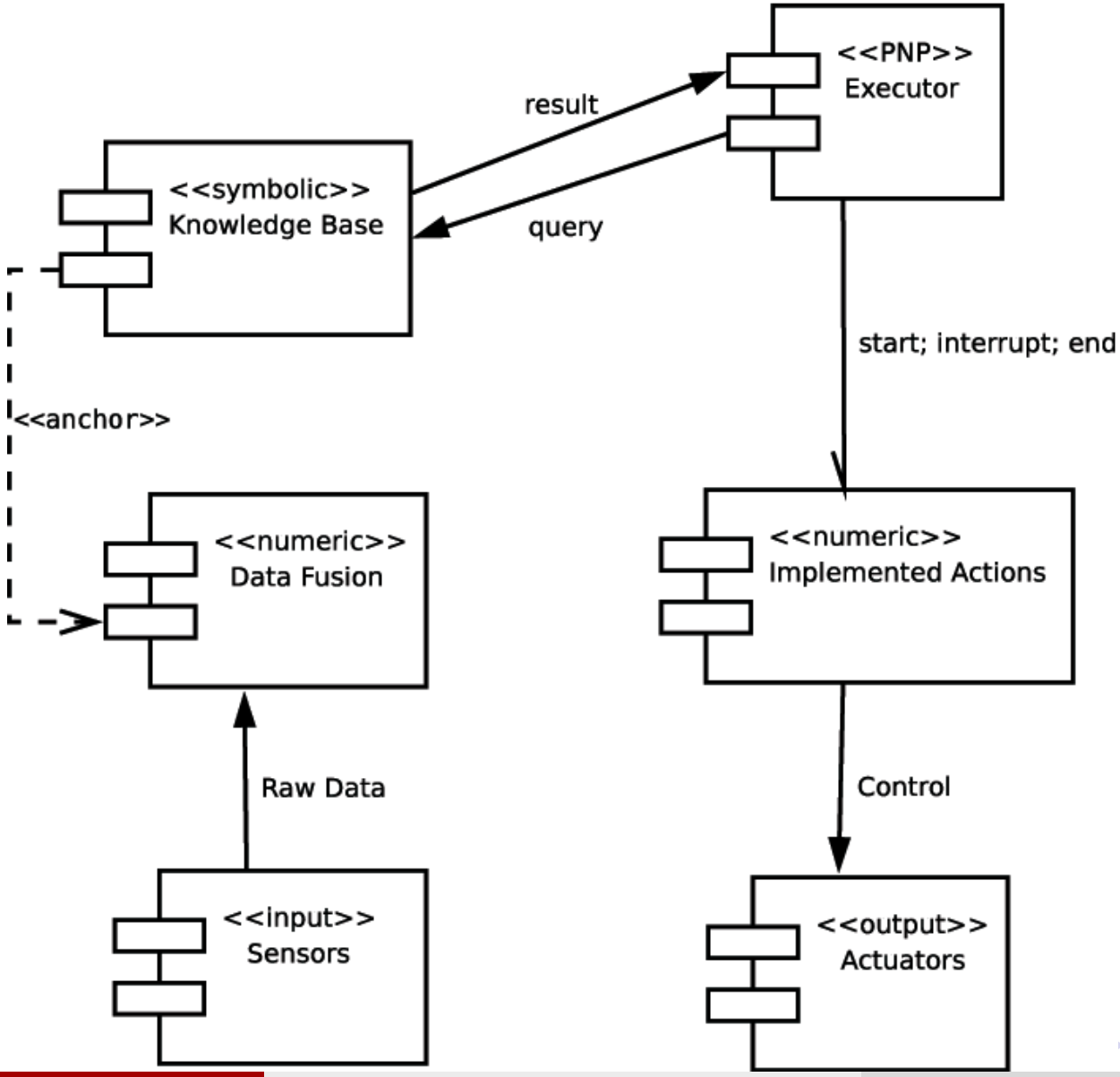
Firing Rule

Definition

1. A transition t is *enabled*, if each input place p_i (i.e. $(p_i, t) \in F$) is marked with at least $w(p_i, t)$ tokens.
2. An enabled transition may or may not fire, depending on whether related event occurs or not.
3. If an enabled transition t fires, $w(p_i, t)$ tokens are removed for each input place p_i and $w(t, p_o)$ are added to each output place p_o such that $(t, p_o) \in F$.

Outline

Abstract Architecture



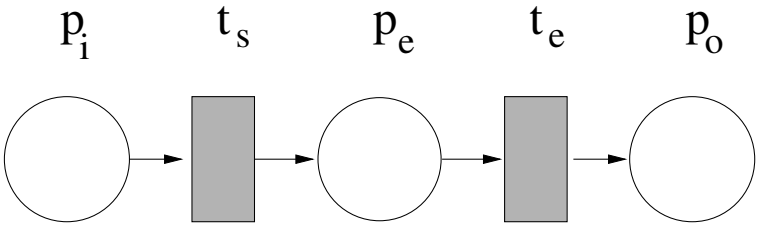
Petri Net Plans [ZI06]

Petri Net Plans

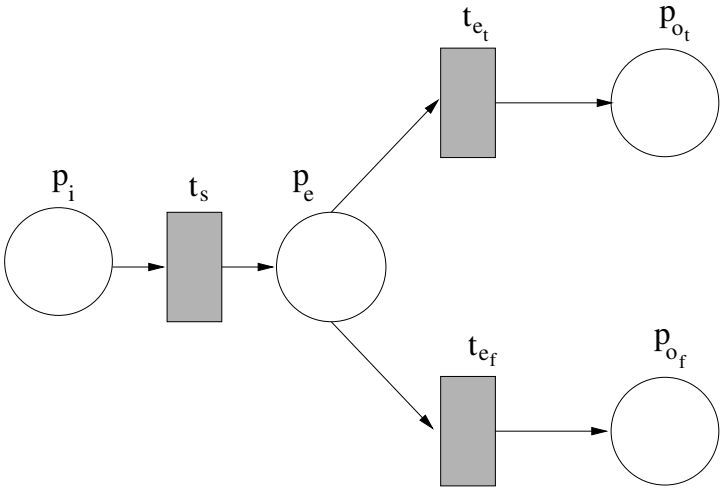
Petri Net Plans are defined in terms of Actions and Operators to combine actions:

- Actions .
  1. ordinary actions.
  2. sensing actions.
- Operators.

Non-Istantaneous Actions



Ordinary Action



Sensing Action

Sequence of Actions

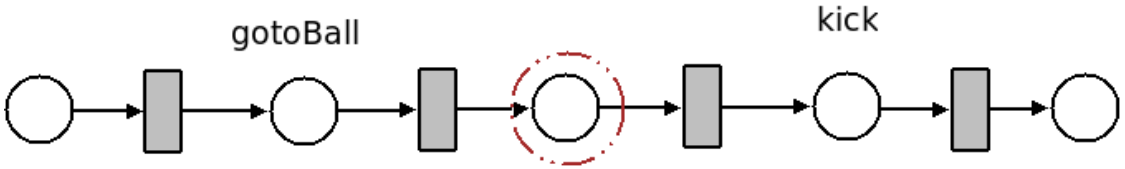
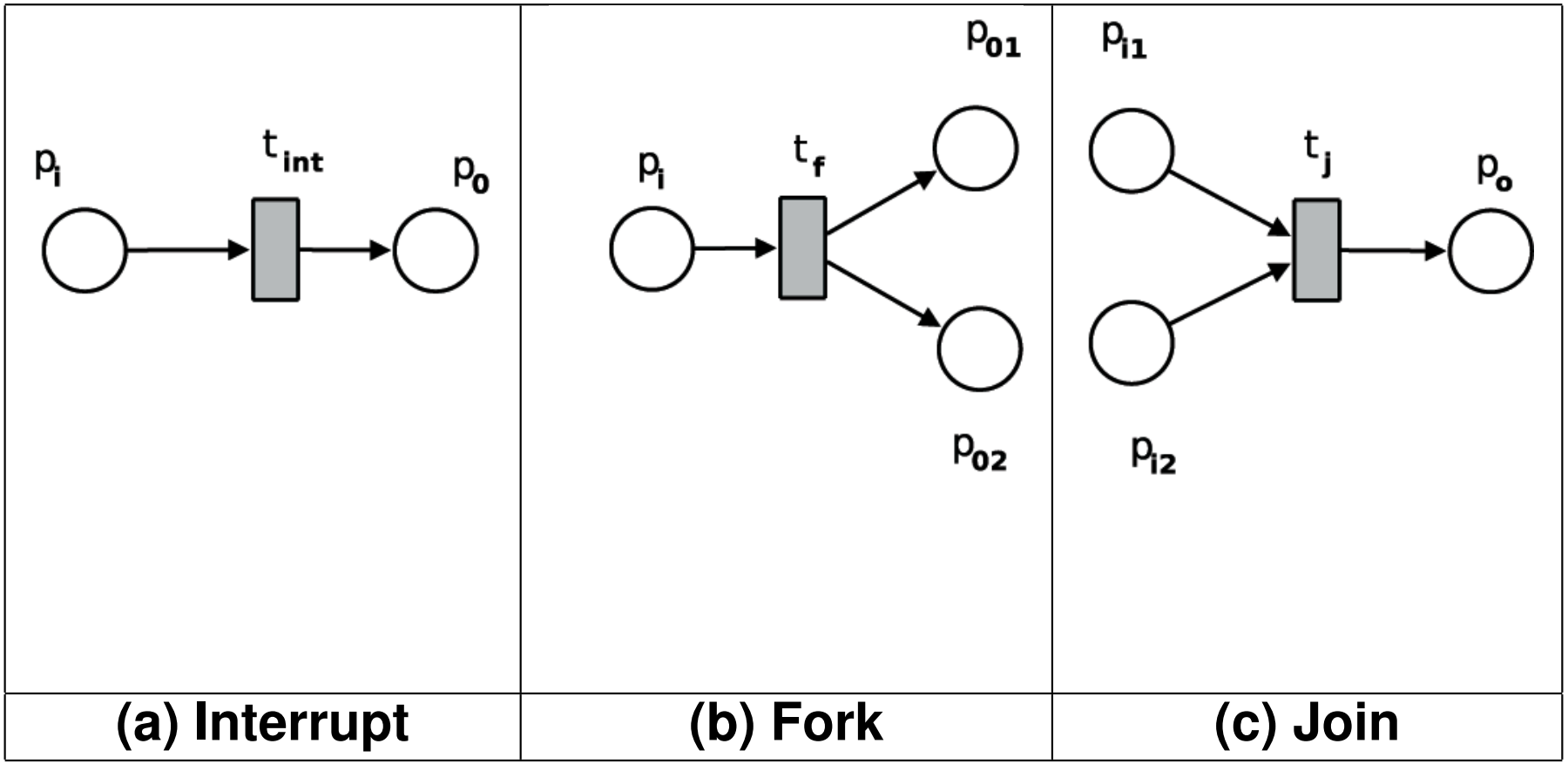


Figure: The sequence of two ordinary actions.

Operators



(a) Interrupt	(b) Fork	(c) Join

Action Interrupt

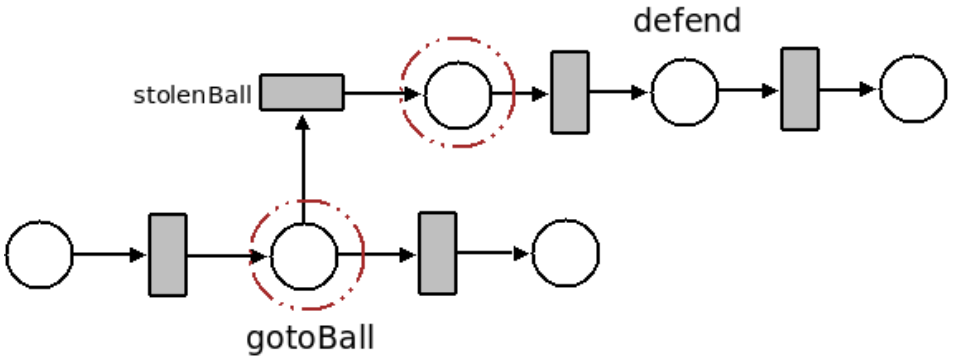


Figure: The interrupt of an action and its recovery procedure.

Concurrency

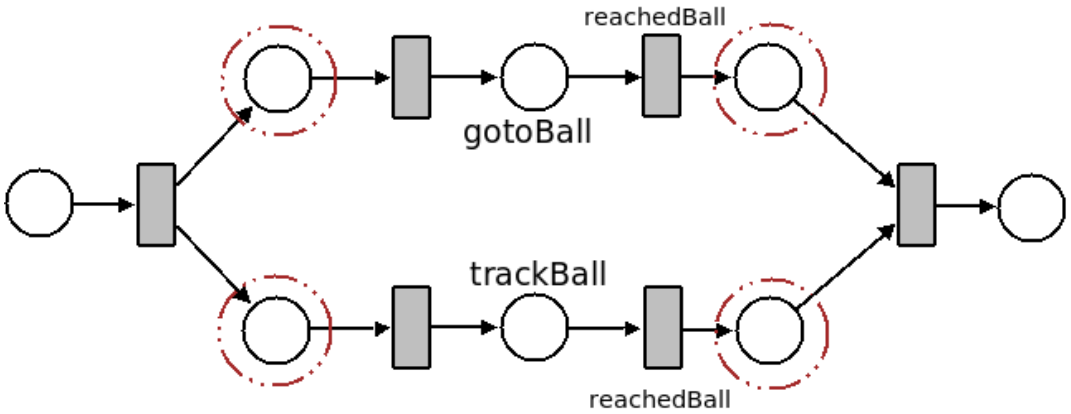


Figure: The fork and the subsequent join of two actions.

Action Implementation

Action Interface

virtual void start();
virtual void end();
virtual void executeStep();
virtual bool finished();

PNP Execution Algorithm

Domains:

$\mathcal{A} = \{a_1, \ldots, a_k\}$: *Set of Implemented actions*
Φ : *Set of terms and formulas about the environment*
$TrType = \{start, end, interrupt, standard\}$

Structures:

$Transition : \langle a \in \mathcal{A}, \phi \in \Phi, t \in TrType \rangle$
$Action : \langle start(), end(), interrupt() \rangle$

Global Variables:

$KnowledgeBase : KB$

PNP Execution Algorithm (con't)

procedure *execute*($PNP\ \langle P, T, F, W, M_0, G\rangle$)

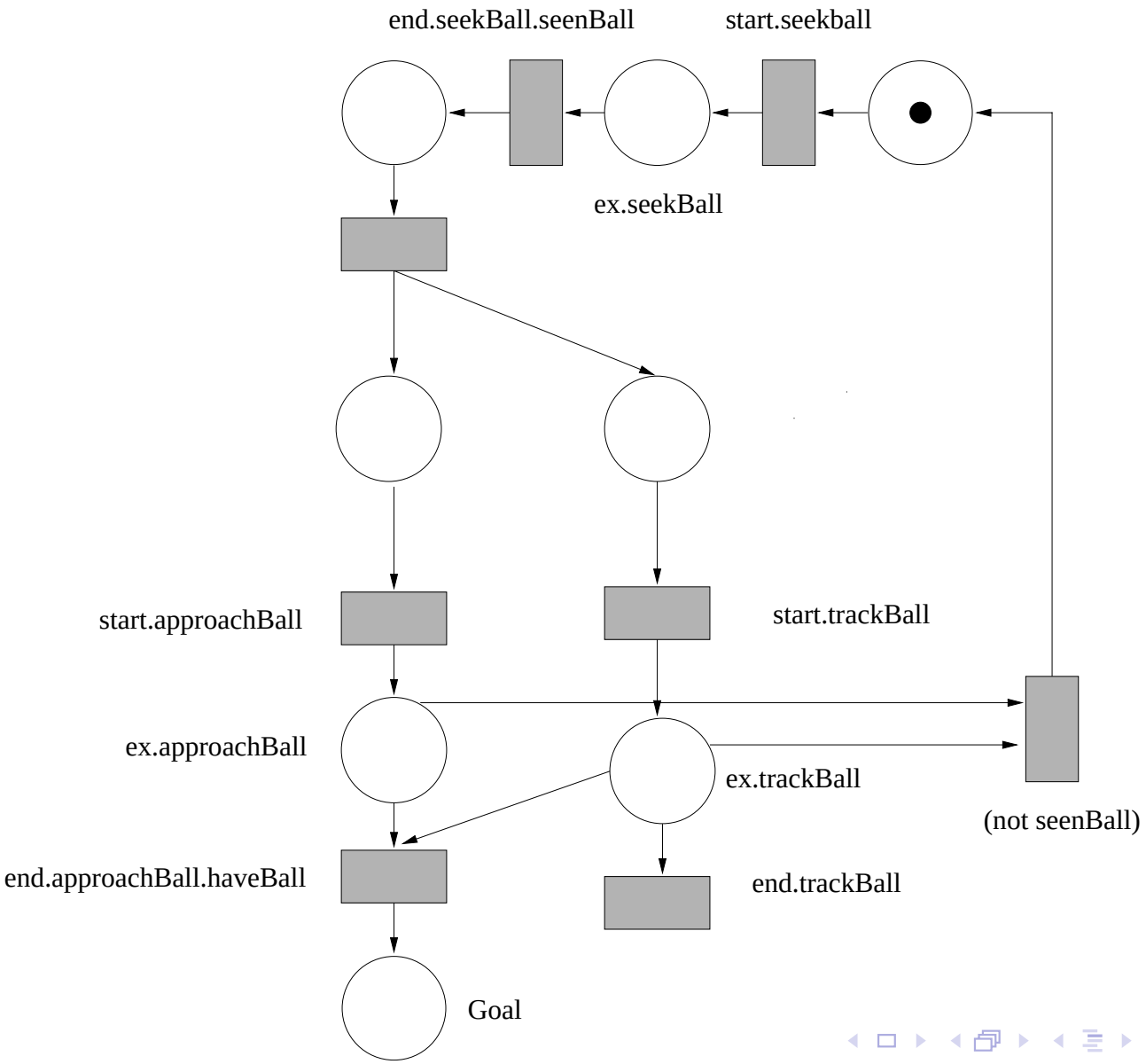
```
CurrentMarking = M_0
while CurrentMarking ∉ G do
    for all t ∈ T do
        if enabled(t) ∧ KB ⊨ t.φ then
            handleTransition(t)
            CurrentMarking = fire(t)
        end if
    end for
end while
```

handleTransition procedure

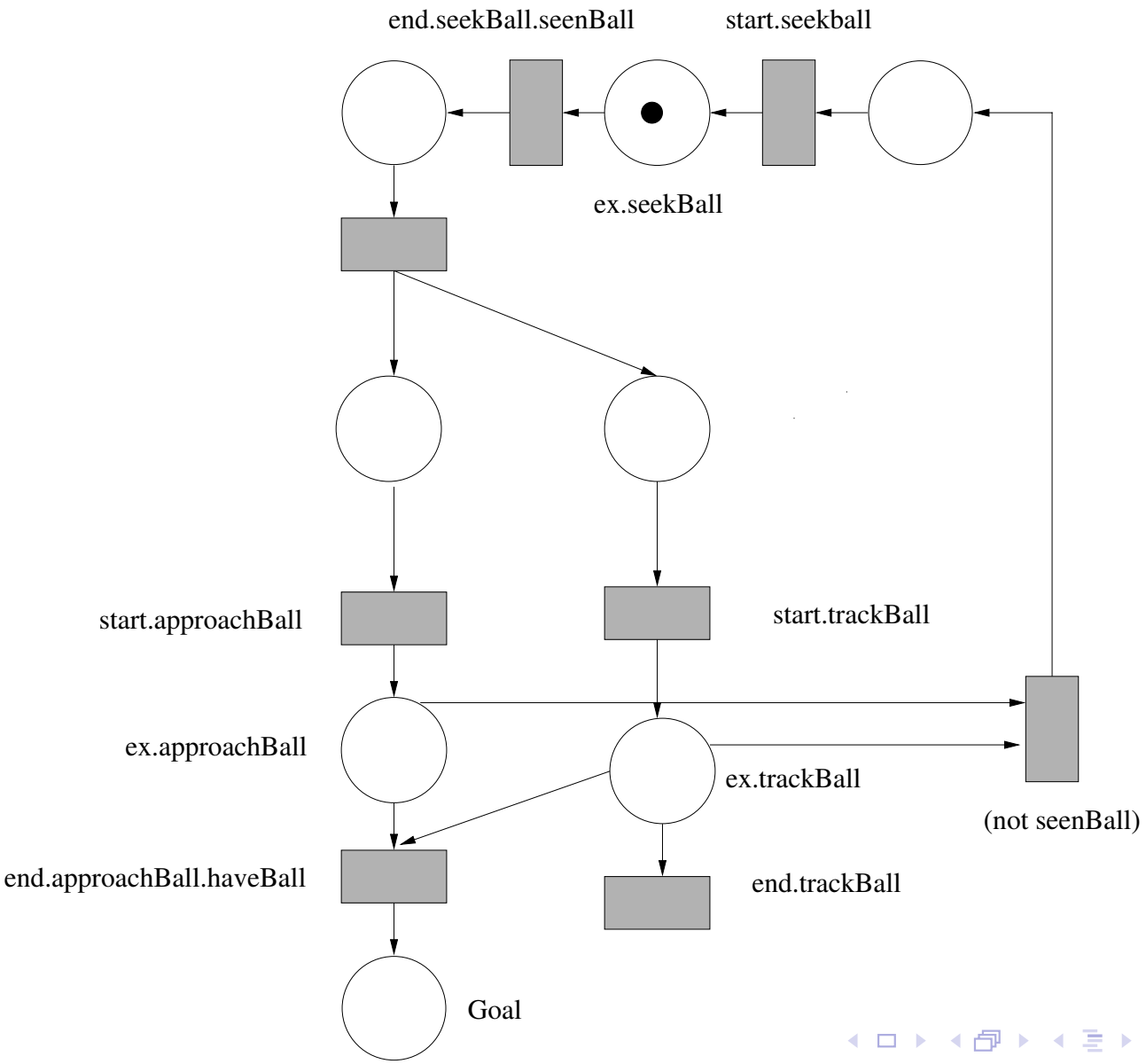
procedure *handleTransition*(t)

  if $t.t = start$ ***then***
    $t.a.start()$
  else if $t.t = end$ ***then***
    $t.a.end()$
  else if $t.t = interrupt$ ***then***
    $t.a.interrupt()$
  end if

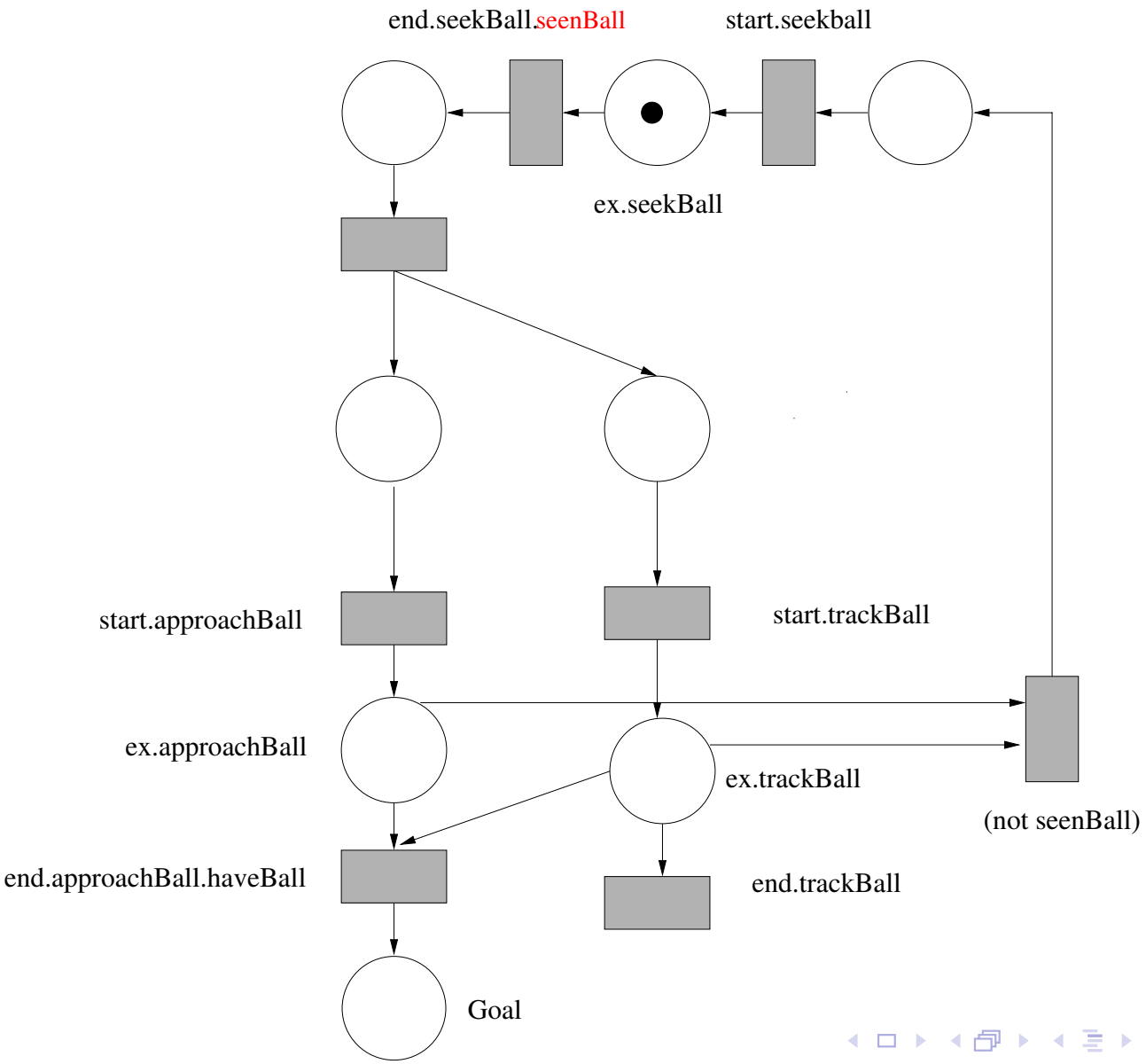
Example: A Simple 4Legged Striker



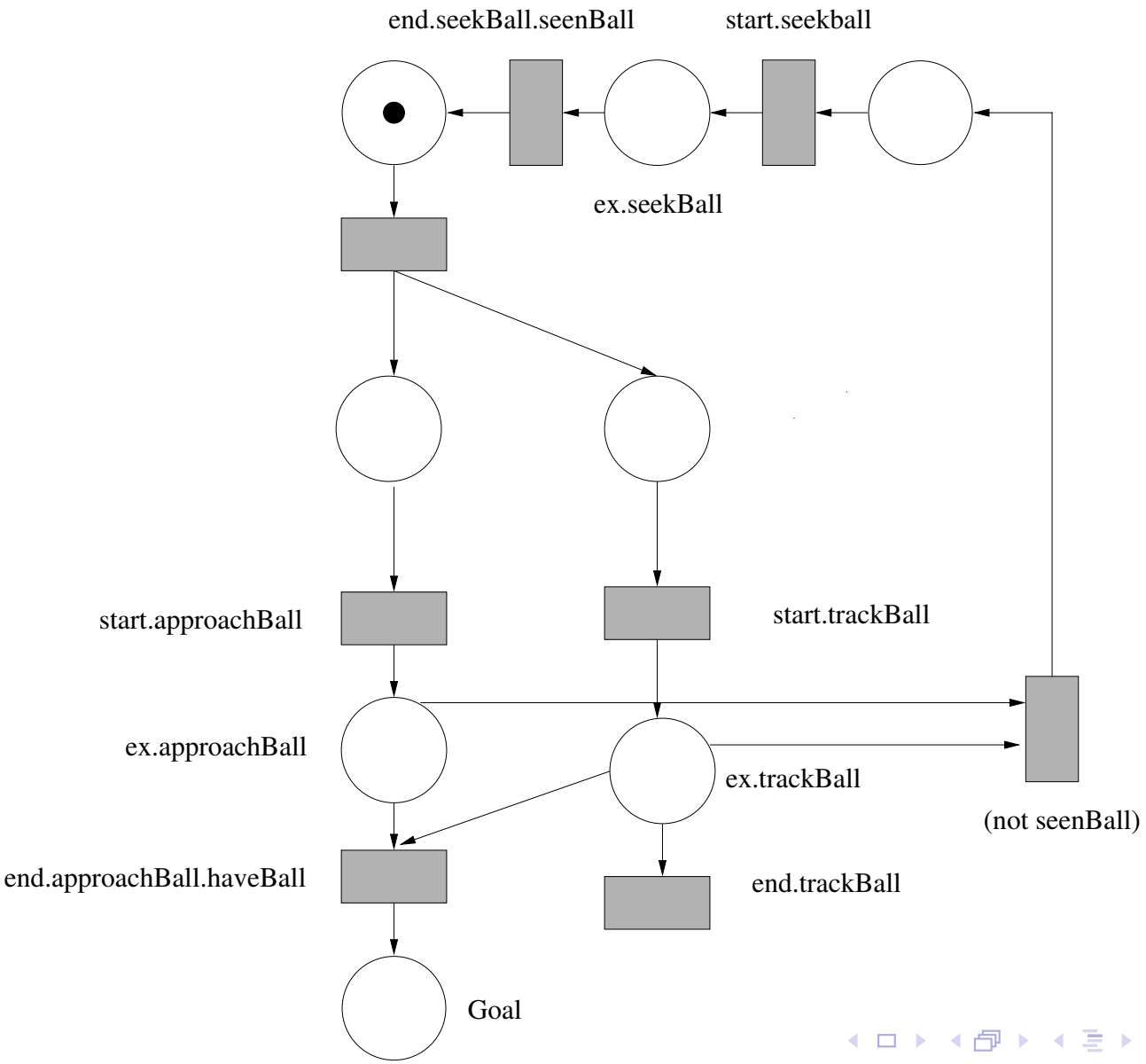
Example: A Simple 4Legged Striker



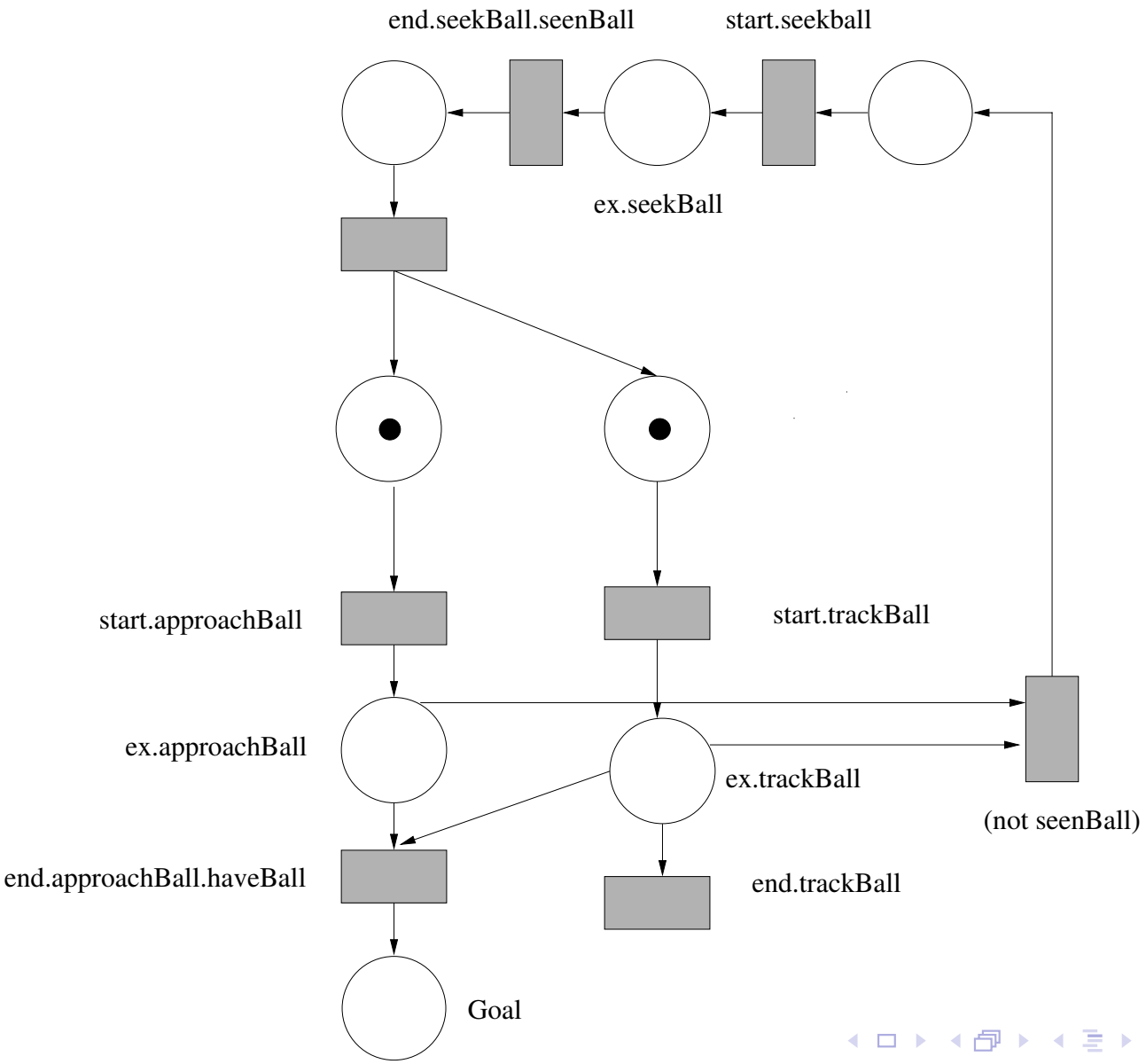
Example: A Simple 4Legged Striker



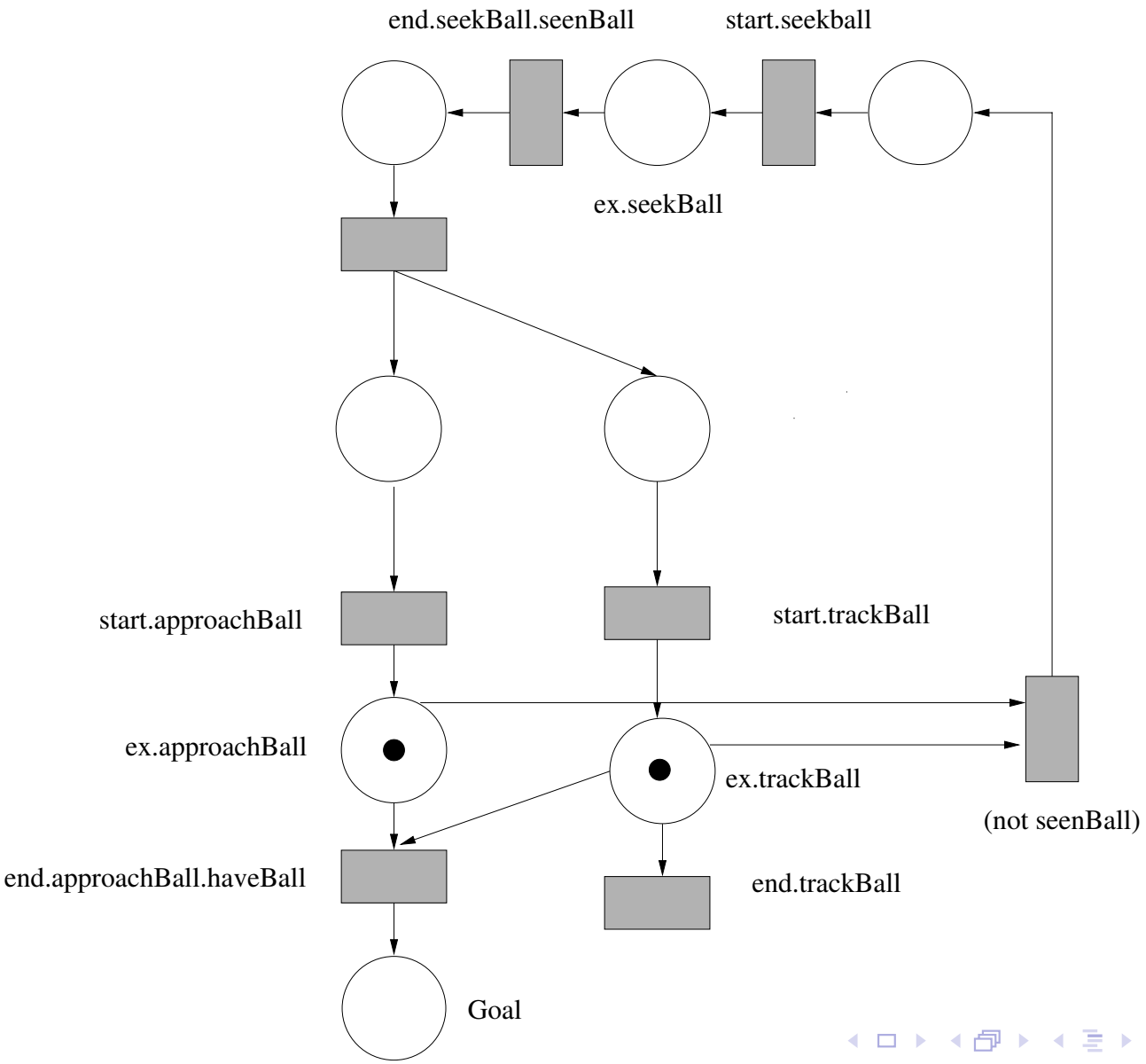
Example: A Simple 4Legged Striker



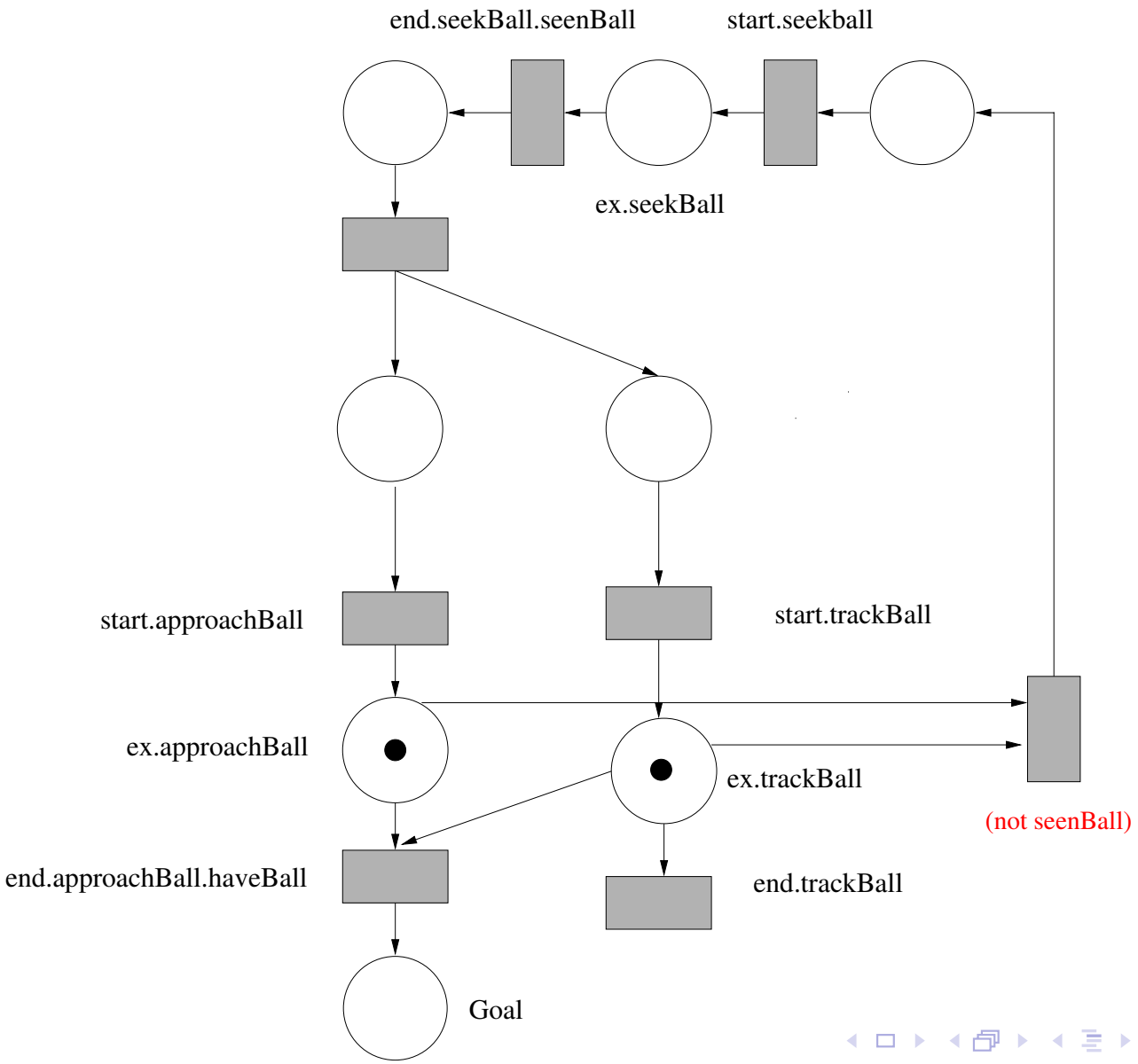
Example: A Simple 4Legged Striker



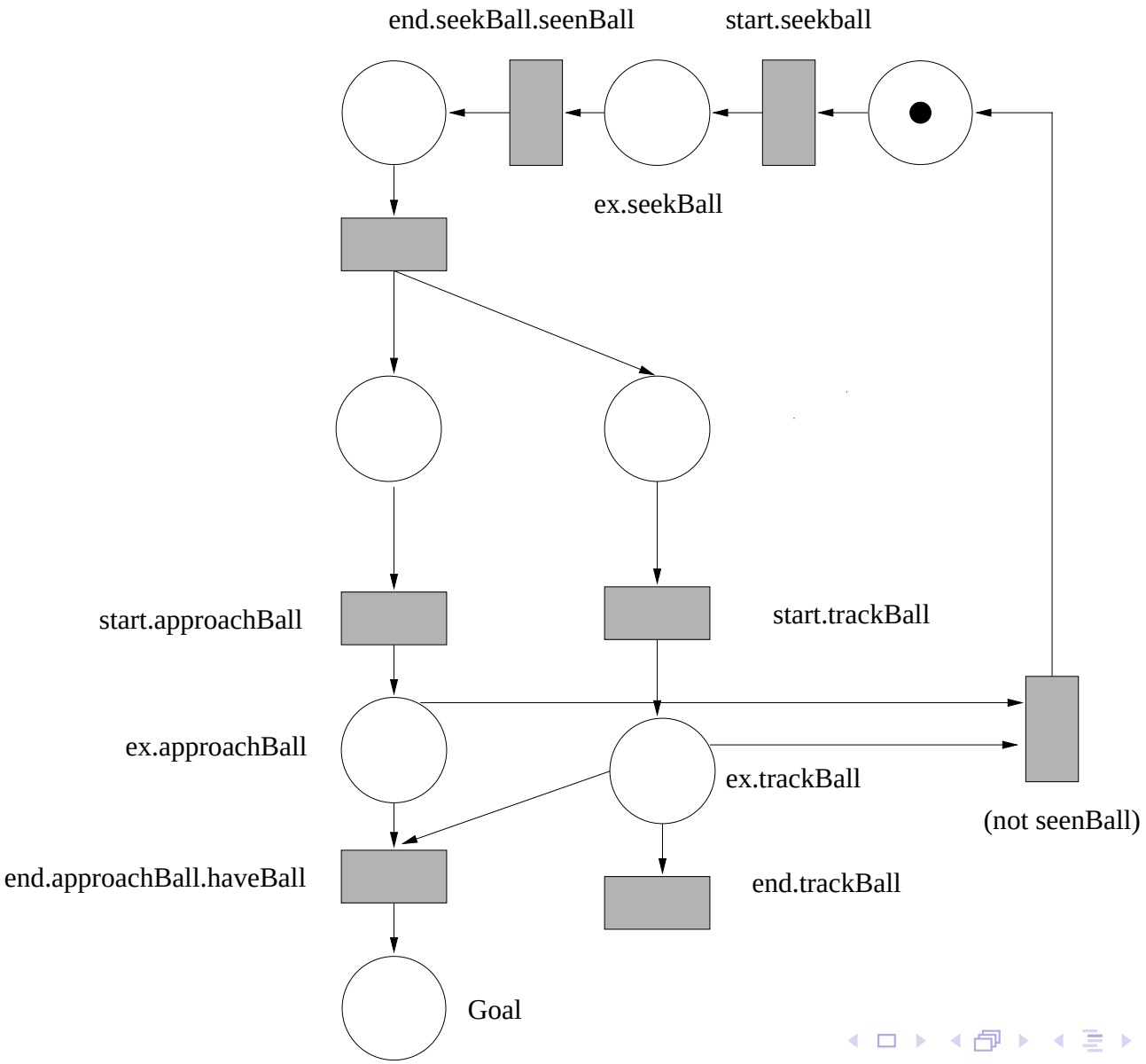
Example: A Simple 4Legged Striker



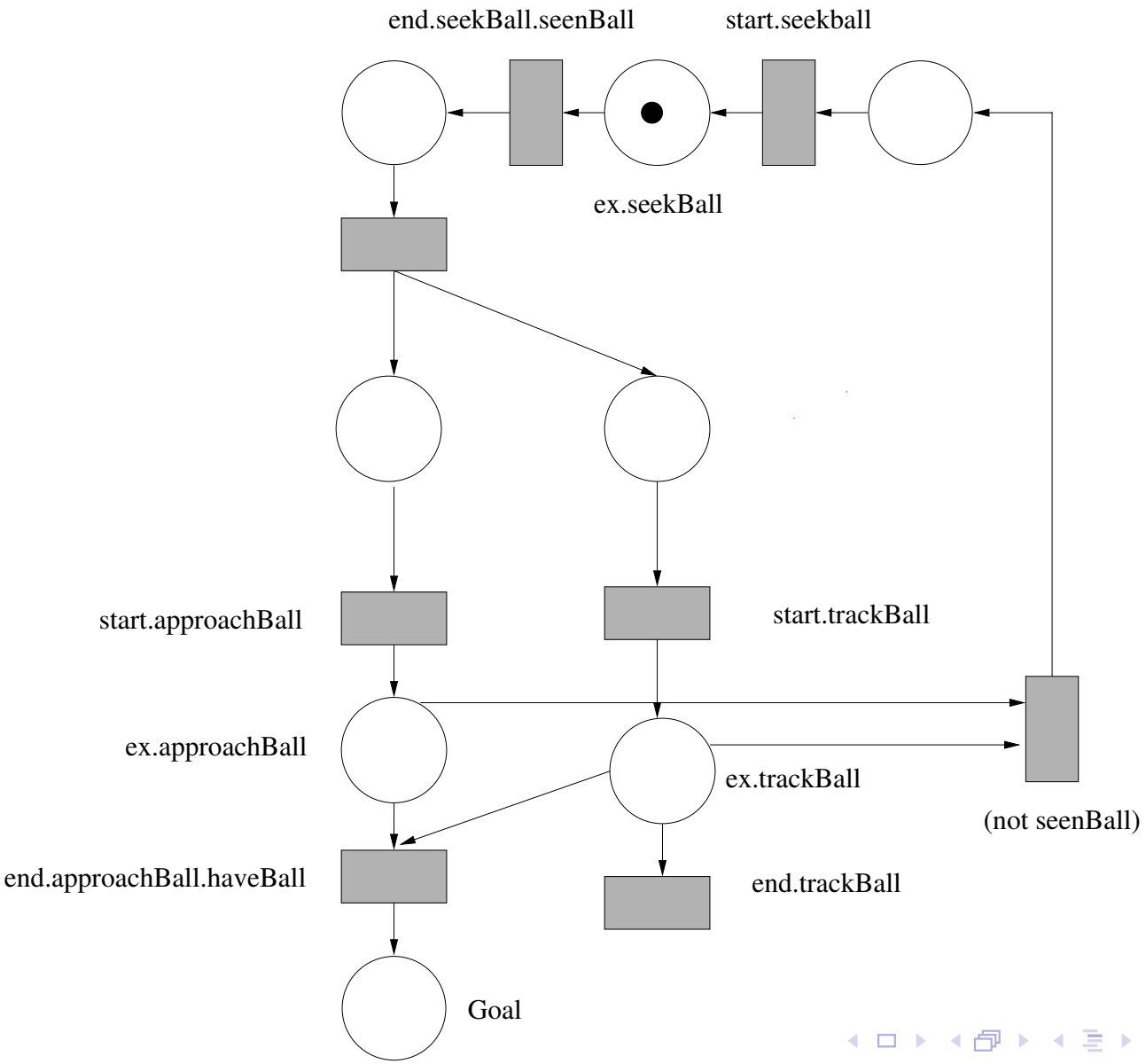
Example: A Simple 4Legged Striker



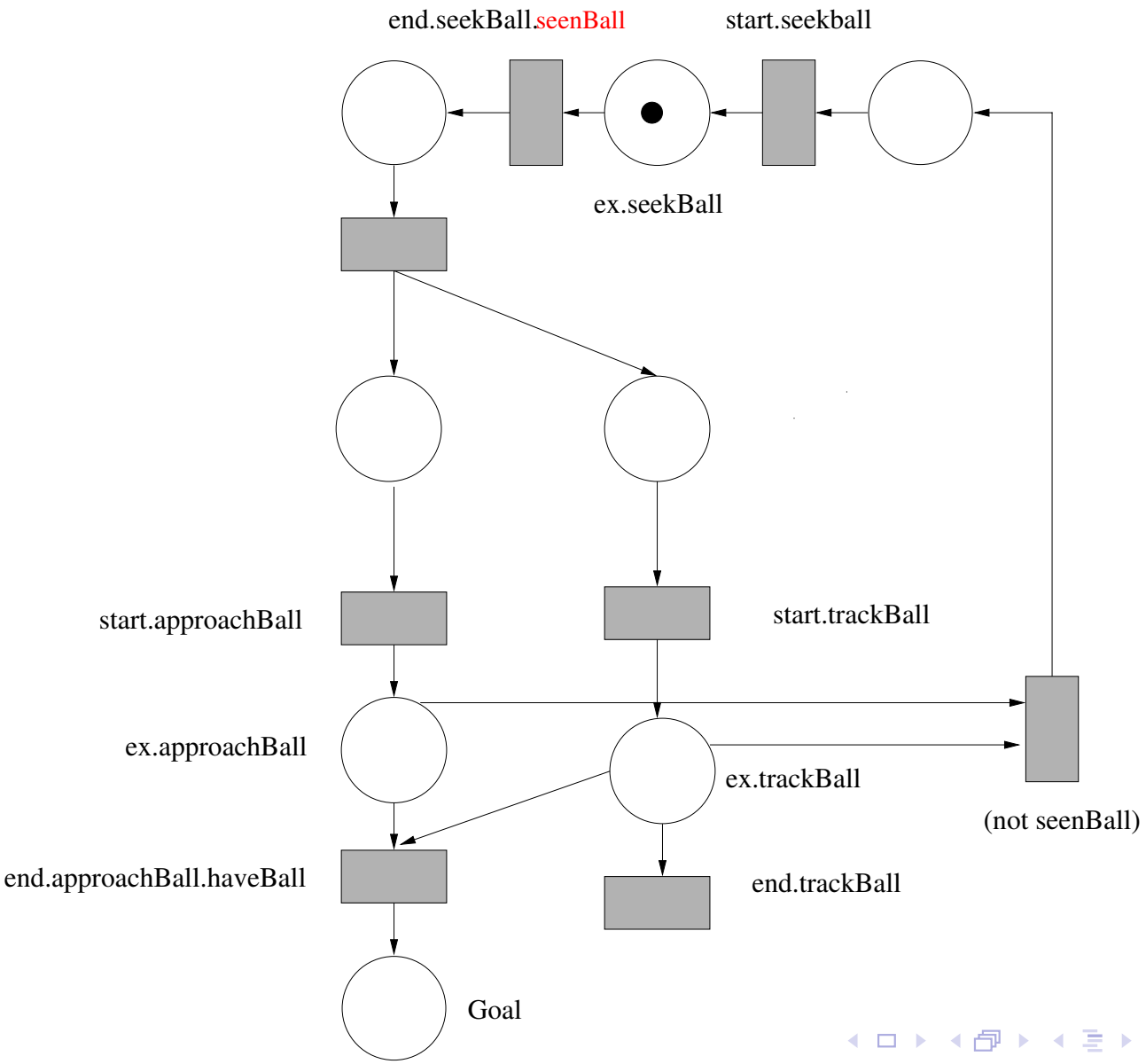
Example: A Simple 4Legged Striker



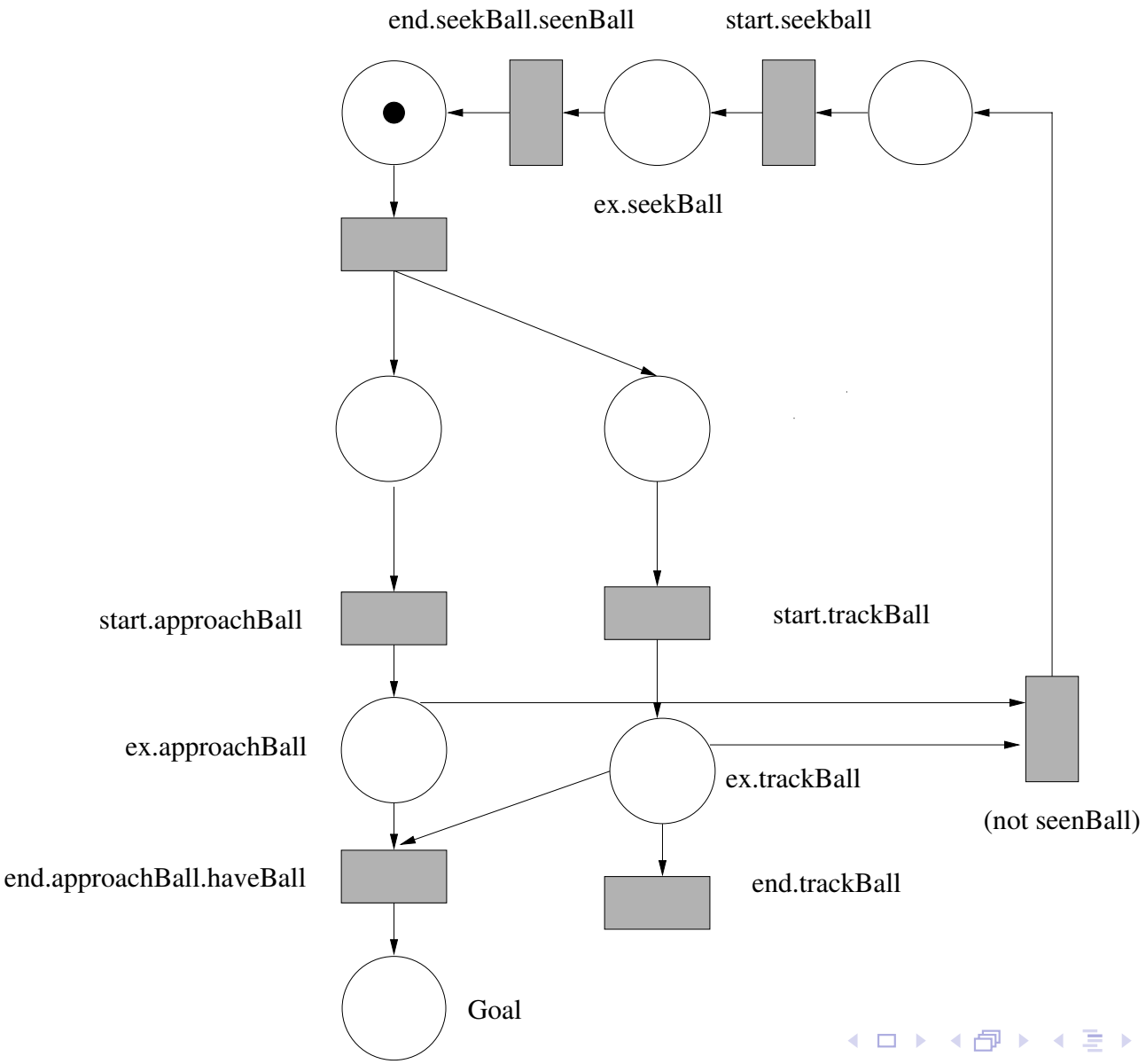
Example: A Simple 4Legged Striker



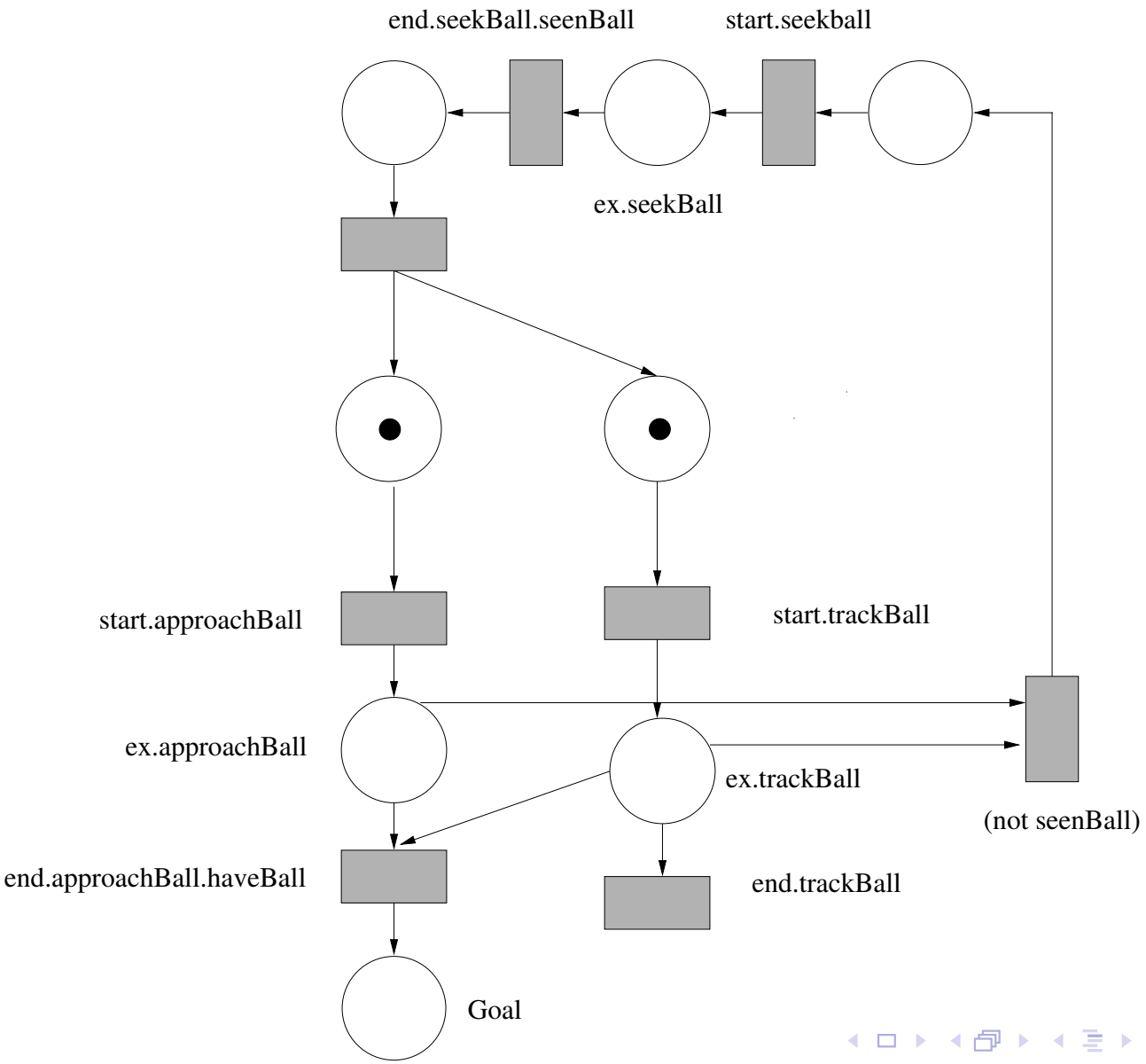
Example: A Simple 4Legged Striker



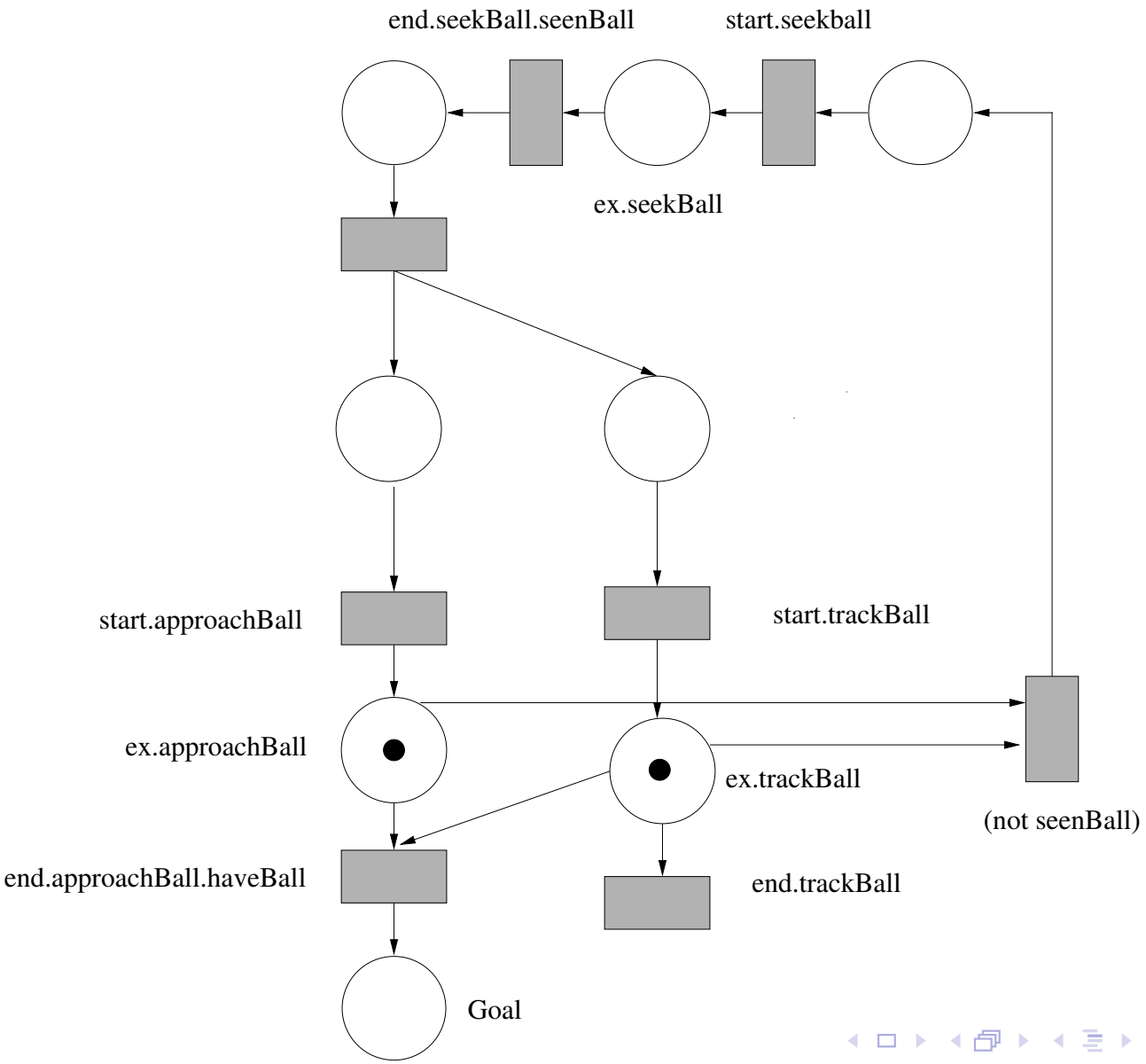
Example: A Simple 4Legged Striker



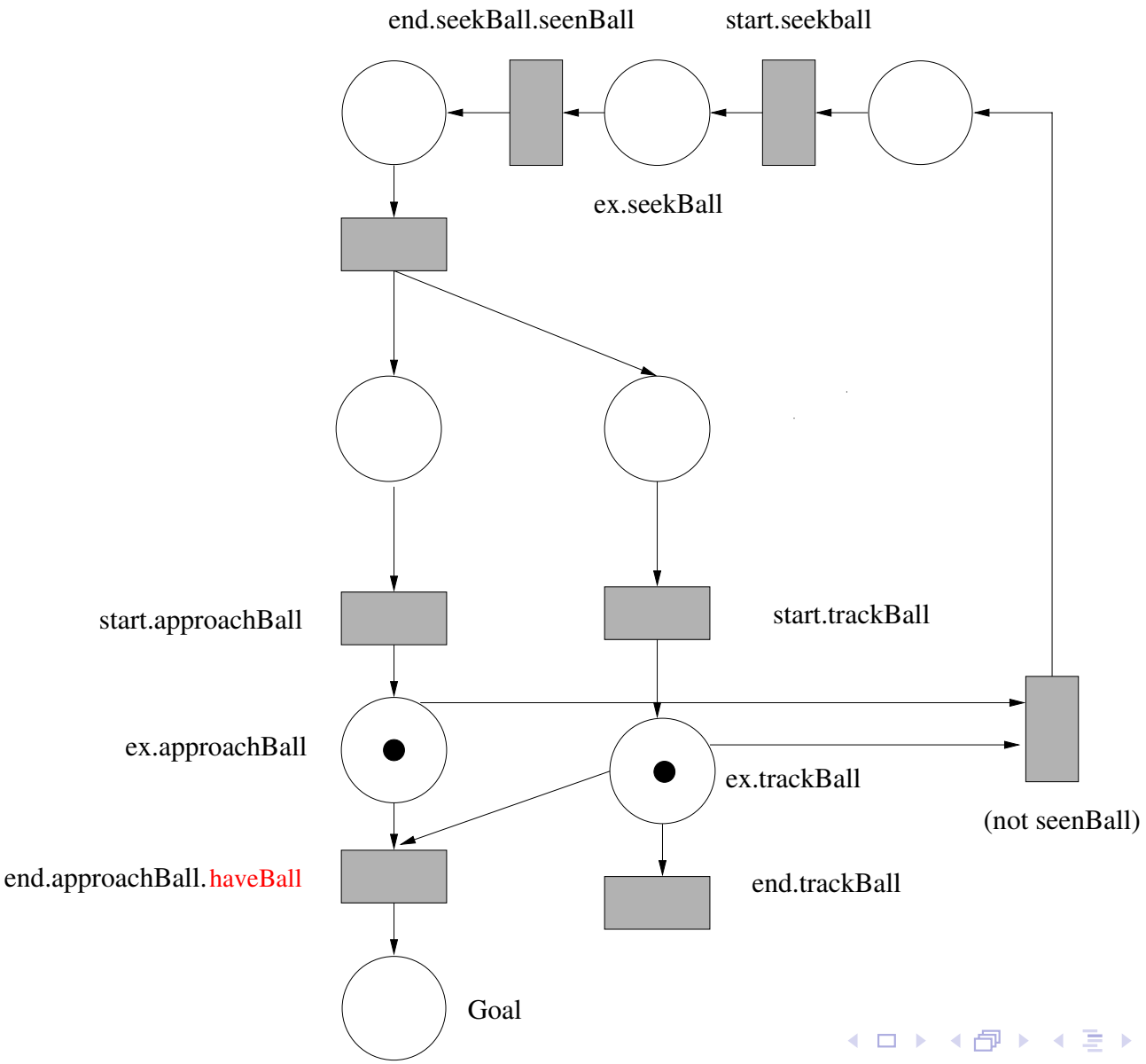
Example: A Simple 4Legged Striker



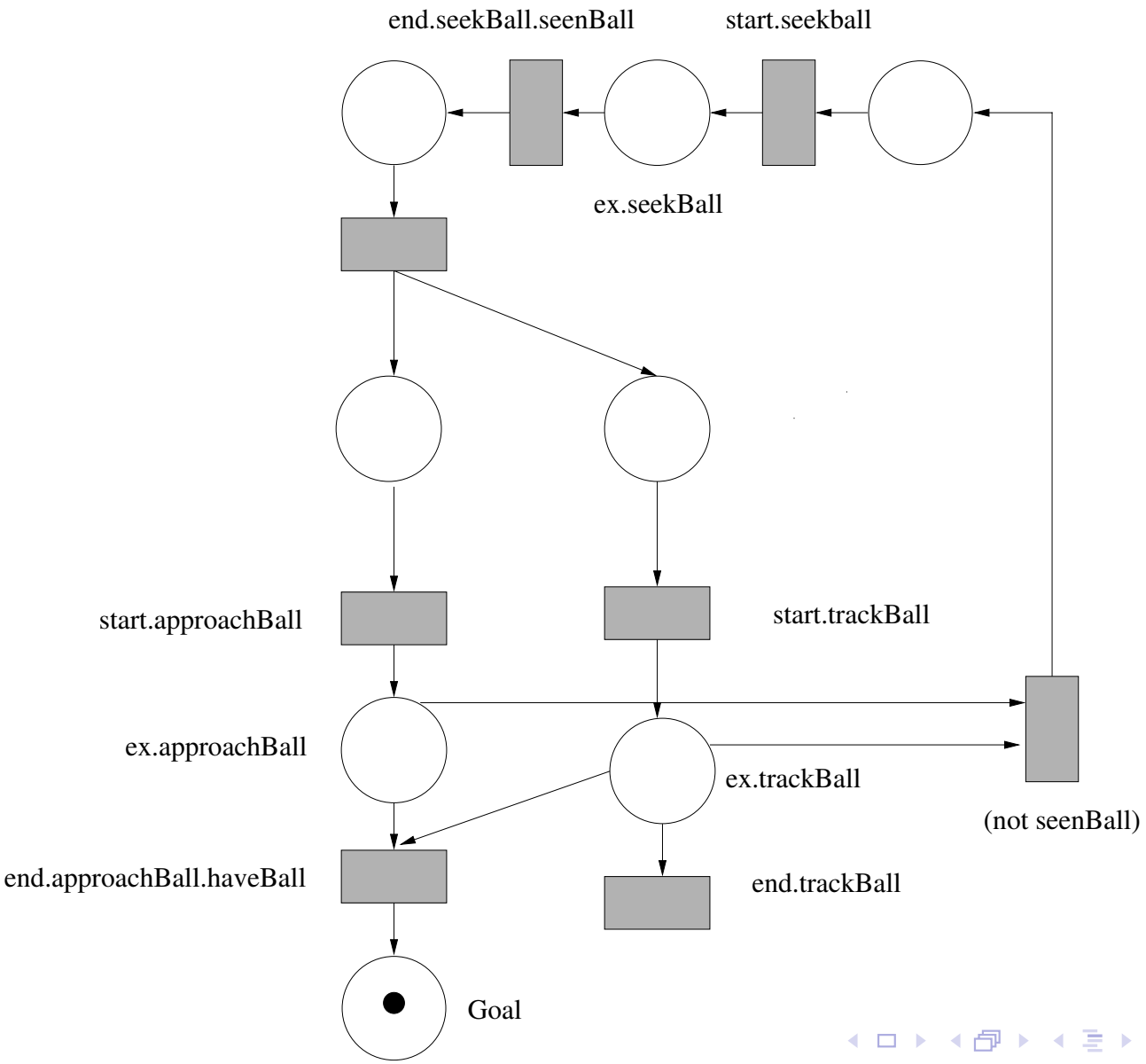
Example: A Simple 4Legged Striker



Example: A Simple 4Legged Striker



Example: A Simple 4Legged Striker



Sub-Plans

- Plans can be organized in a hierarchy, allowing for modularity and reuse
- Sub-plans are like actions
  - When started the initial marking is set
  - Ends when goal marking is reached

Correctness of the Execution Algorithm

In a Nutshell

1. Provide an operational semantics based on Petri nets and the robot's local knowledge (KB_i).
2. Prove that the execution algorithm is correct w.r.t. the operational semantics.

Theorem

[ZI06] If a PNP can be correctly executed, then the Execution Algorithm computes a sequence of transitions $\{M_0, ..., M_n\}$*, such that* M_0 *is the initial marking,* M_n *is a goal marking, and* $M_i \Rightarrow M_{i+1}$*, for each* $i = 0, \ldots, n-1$*.*

Implemented Systems

Robotic Soccer [INMZ06]

Search and Rescue [CFIN07]

Outline

Centralized Plans for Distributed Execution

Definition

[. . .] a partial order planner can generate plans where there need not be a strict ordering between some actions, and in fact where those actions can be executed in parallel. A centralized coordinator agent with such a plan can break it into separate threads, possibly with some synchronization actions. [Dur99].

Multi-Robot PNP

Definition

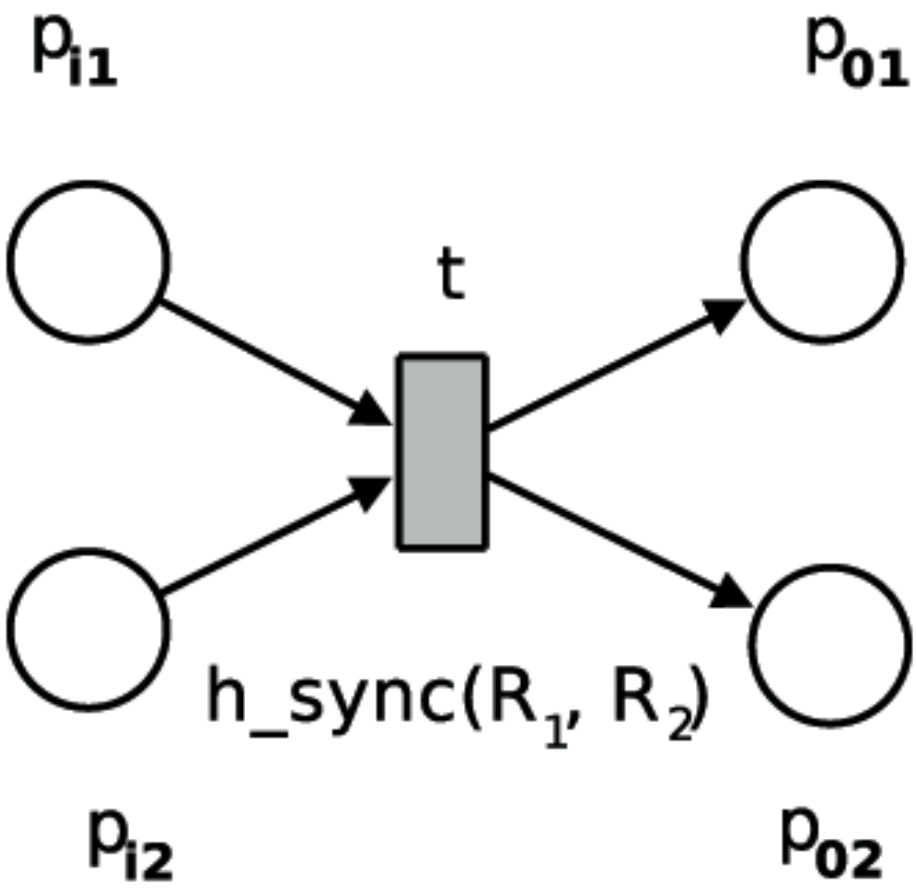
A collection of single-robot PNPs synchronized through the use of multi-robot operators:

- Hard Synchronization
- Soft Synchronization
- Multi-Robot Interrupt

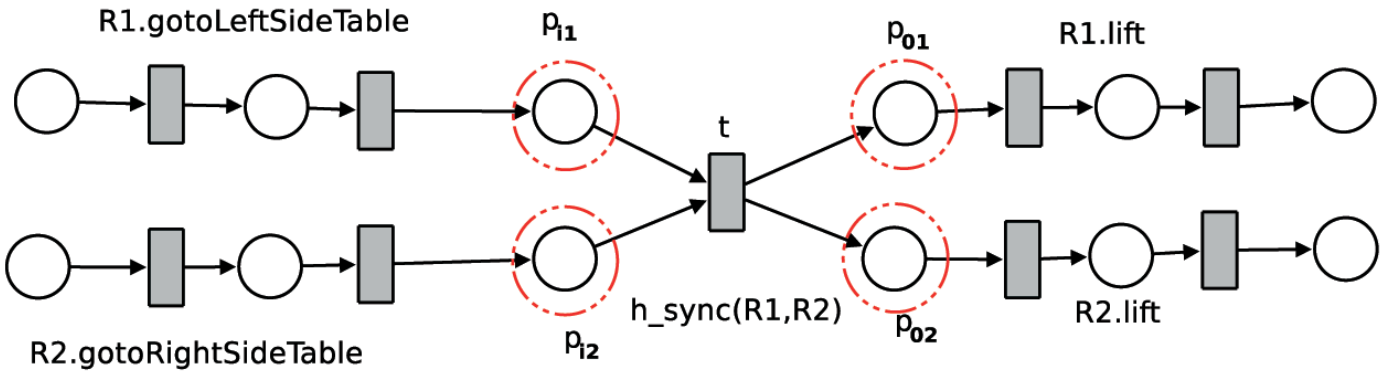
The Idea

Decompose multi-robot operators into single-robot communication primitives to allow distributed execution.

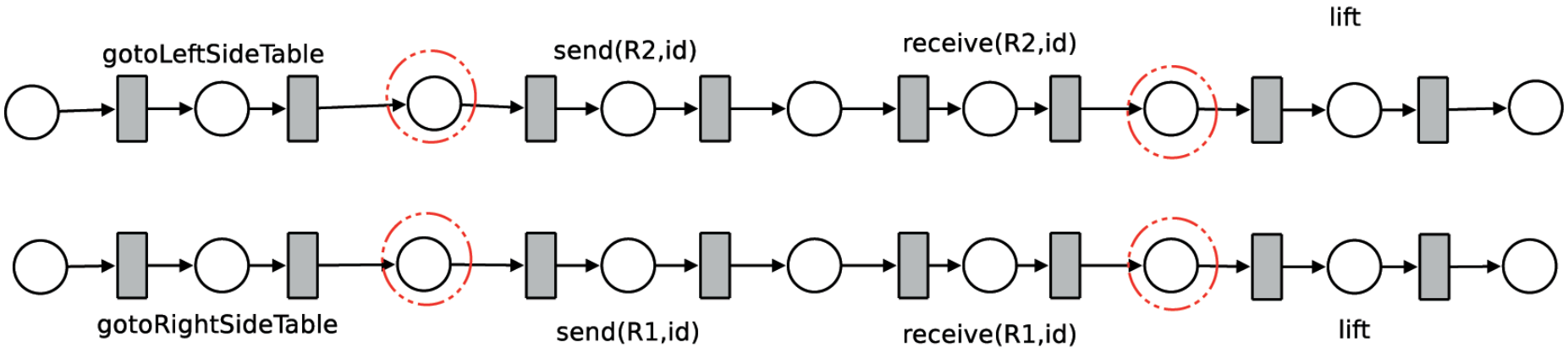
Hard Synchronization



Hard Synchronization Decomposition

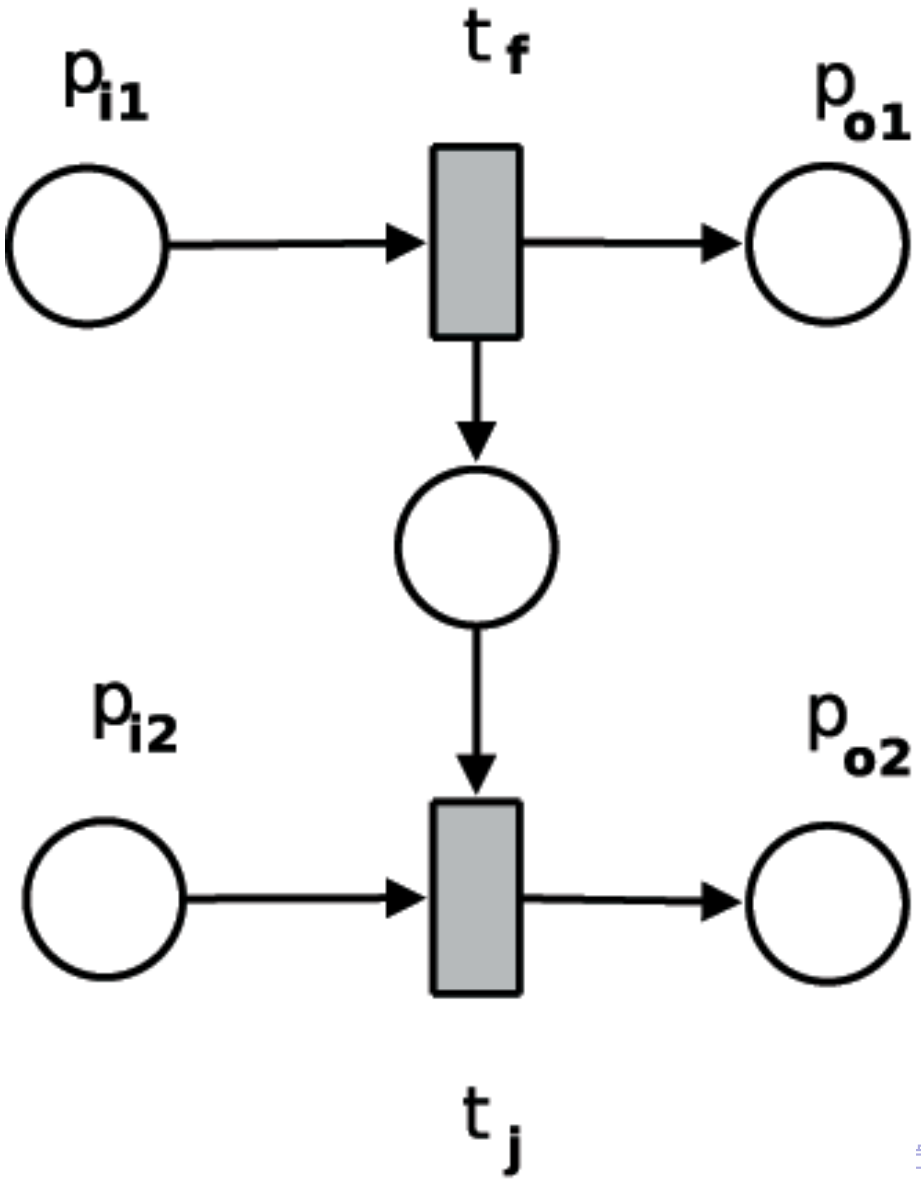


(a)

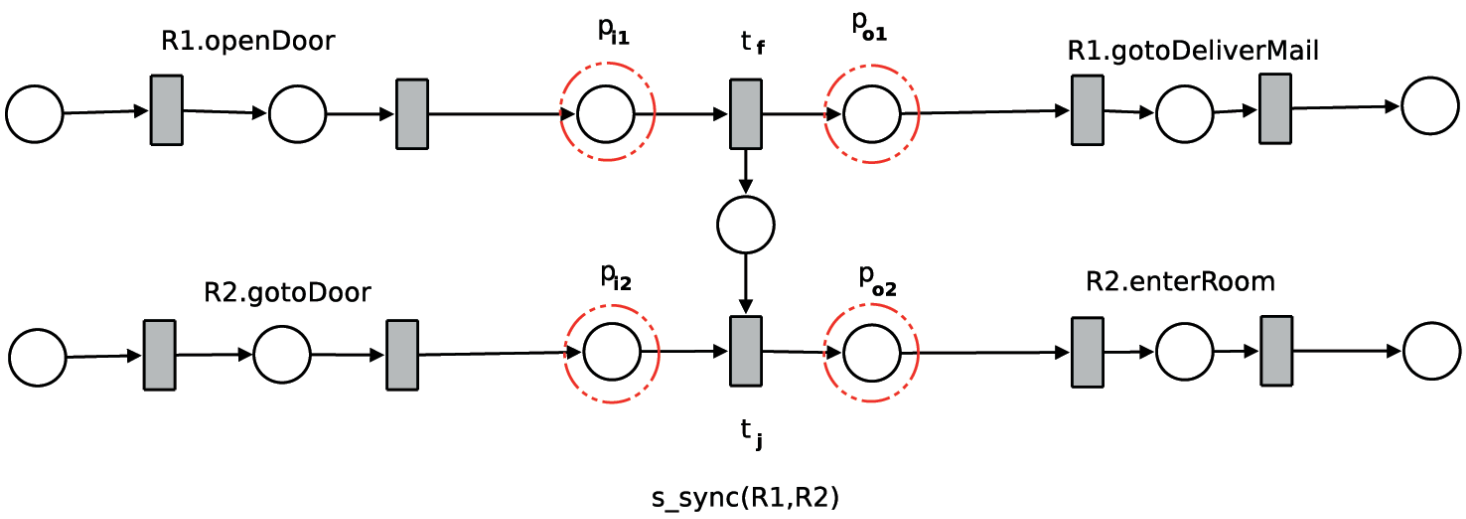


(b)

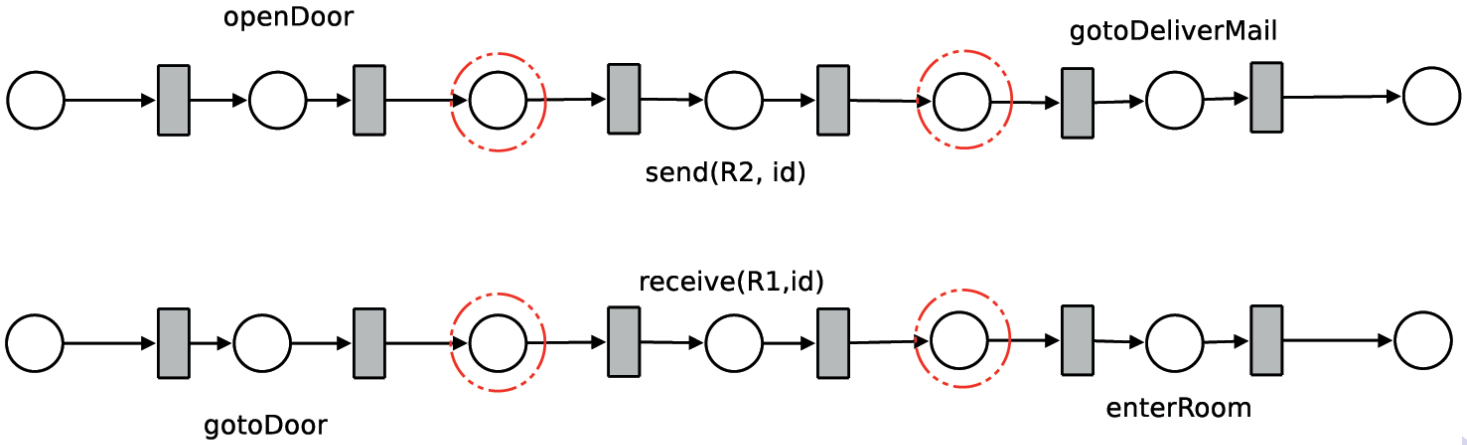
Soft Synchronization



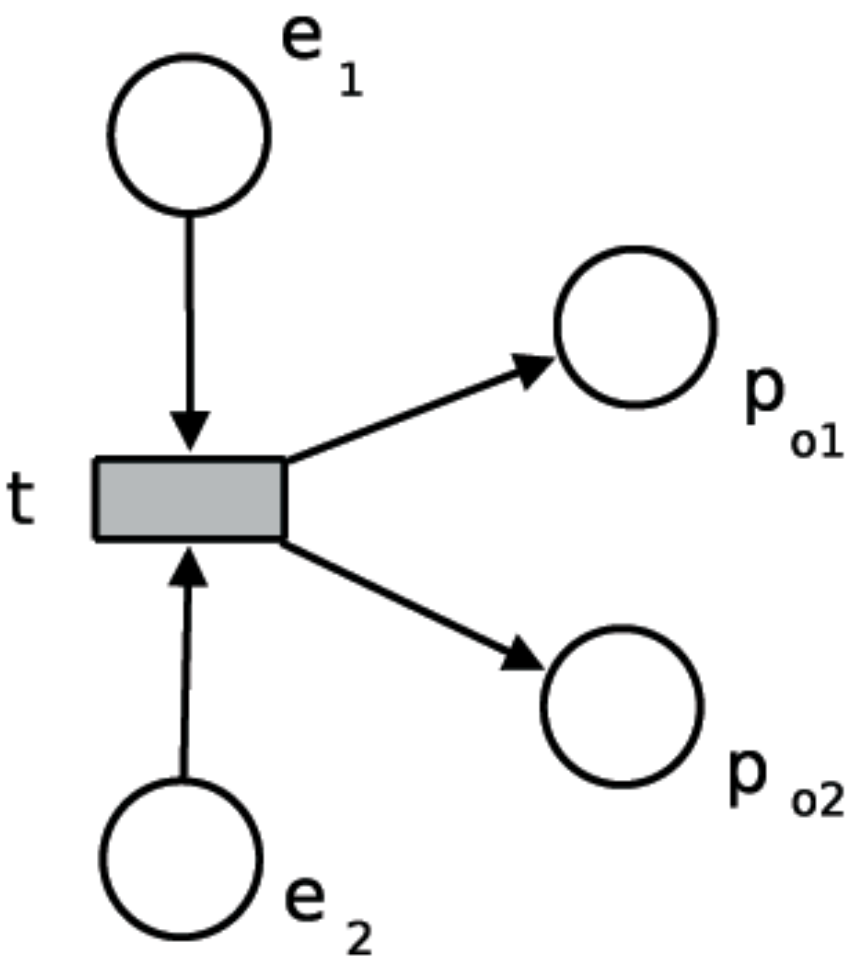
Soft Synchronization Decomposition



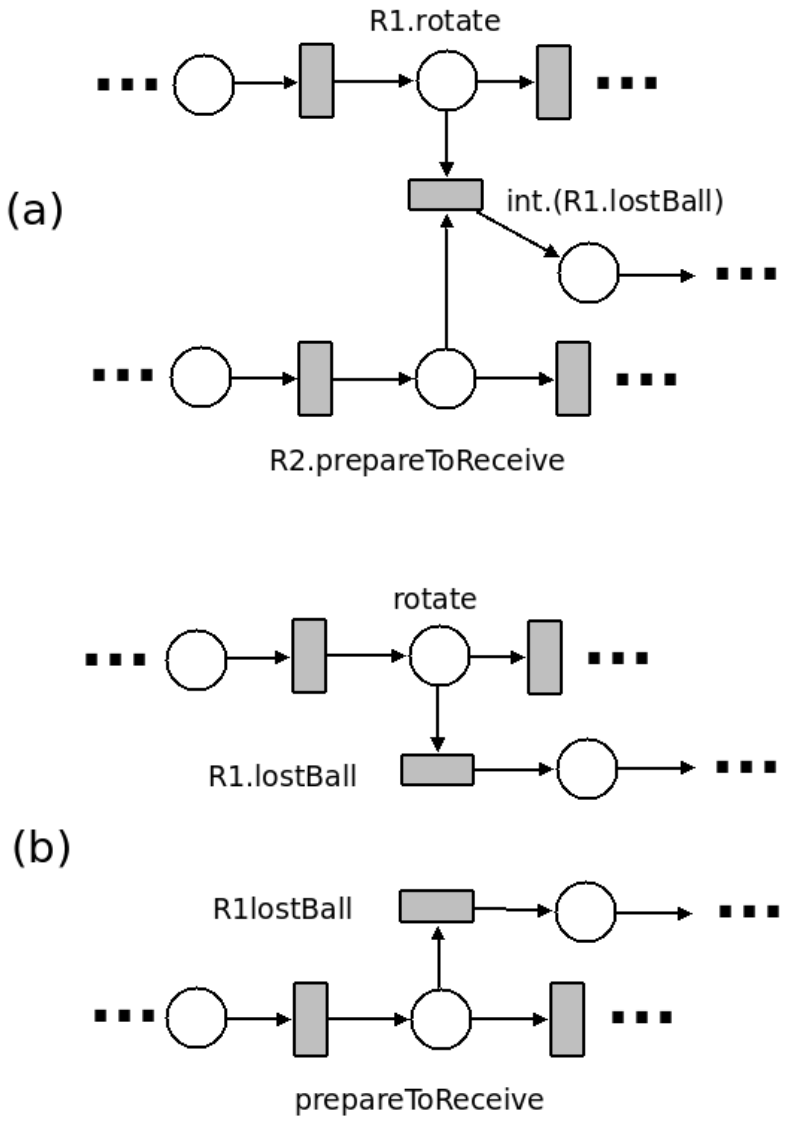
(a)



Multi-Robot Interrupt



Multi-Robot Interrupt Decomposition



Correctness of Distributed Execution

Definition

The distributed execution of a multi-robot PNP is correct iff it enforces the same ordering constraints of its centralized model.

Theorem

The distributed execution of multi-robot PNPs is correct.

Implemented Systems

Ball Passing [PZI+08]

Foraging [FINZ06]

Outline

Conclusions

Discussion

1. Efficient an intuitive framework for designing, writing, executing, and debugging multi-robot plans.
2. Distributed Execution.
3. Suitable formal analysis based on standard PN tools.
4. Thoroughly tested and implemented in different scenarios/platforms.

Future Work

1. Provide a formal description of actions using some action specification language.
2. Implement verification and plan assistant tools in order to guarantee the safeness of plans.

The End

Any Questions?

D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi.
Multi-objective exploration and search for autonomous rescue robots.
Journal of Field Robotics, Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems, 24:763–777, August - September 2007.

H. Costelha and P. Lima.
Modelling, analysis and execution of robotic tasks using petri nets.
In *Proceeding of Interantional Conference on Intelligent Robots and Systems (IROS)*, pages 1449–1454, 2007.

Edmund H. Durfee.
Distributed problem solving and planning.
In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press, 1999.

A. Farinelli, L. Iocchi, D. Nardi, and V. A. Ziparo.

Assignment of dynamically perceived tasks by token passing in multi-robot systems.
Proceedings of the IEEE, Special issue on Multi-Robot Systems, 94(7):1271–1288, 2006.
ISSN:0018-9219.

L. Iocchi, D. Nardi, L. Marchetti, and V. A. Ziparo.
S.P.Q.R. Legged Team Description Paper.
RoboCup Symposium 2006 (CD-ROM Proceedings), 2006.

K. Konolige.
COLBERT: A language for reactive control in saphira.
Lecture Notes in Computer Science, 1303:31–50, 1997.

Martin Lötzsch, Joscha Bach, Hans-Dieter Burkhard, and Matthias Jüngel.
Designing agent behavior with the extensible agent behavior specification language XABSL.
In Daniel Polani, Brett Browning, and Andrea Bonarini, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of

Lecture Notes in Artificial Intelligence, pages 114–124, Padova, Italy, 2004. Springer.

P.F. Palamara, V.A. Ziparo, L. Iocchi, D. Nardi, P. Lima, and H. Costelha.
A robotic soccer passing task using Petri Net Plans (demo paper).
In Parkes Müller Padgham and Parsons, editors, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 1711–1712, Estoril, Portugal, May 2008. IFAAMAS Press.

Weihua Sheng and Qingyan Yang.
Peer-to-peer multi-robot coordination algorithms: petri net based analysis and design.
Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on, pages 1407–1412, 2005.

V. A. Ziparo and L. Iocchi.
Petri net plans.

In *Proceedings of Fourth International Workshop on Modeling of Objects, Components, and Agents (MOCA)*, pages 267–290, Turku, Finland, 2006.
Bericht 272, FBI-HH-B-272/06.