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

1. Introduction
2. Petri Nets in a Nutshell
3. Basic PNPs
4. Multi-Robot
5. Conclusions
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
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

1. Introduction
2. Petri Nets in a Nutshell
3. Basic PNPs
4. Multi-Robot
5. Conclusions
Petri Nets in a Nutshell

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

1. Introduction
2. Petri Nets in a Nutshell
3. Basic PNPs
4. Multi-Robot
5. Conclusions
Abstract Architecture

- Knowledge Base
- Data Fusion
- Sensors
- Executors
- Implemented Actions
- Raw Data
- Control

Relations:
- Start; interrupt; end
- Result
- Query
Petri Net Plans [ZI06]

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

$p_i \rightarrow t_s \rightarrow p_e \rightarrow t_e \rightarrow p_o$

$t_{ef} \rightarrow p_{of}$
**Sequence of Actions**

Figure: The sequence of two ordinary actions.

![Diagram showing the sequence of actions: gotoBall followed by kick.]
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();
Basic PNPs

PNP Execution Algorithm

Domains:

\[ A = \{ a_1, \ldots, a_k \} : \text{Set of Implemented actions} \]
\[ \Phi : \text{Set of terms and formulas about the environment} \]
\[ \text{TrType} = \{ \text{start}, \text{end}, \text{interrupt}, \text{standard} \} \]

Structures:

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

Global Variables:

\[ \text{KnowledgeBase} : KB \]
**PNP Execution Algorithm (con’t)**

```plaintext
procedure execute(PNP \( \langle P, T, F, W, M_0, G \rangle \))

1: CurrentMarking = \( M_0 \)
2: while CurrentMarking \( \not\in G \) do
3:   for all \( t \in T \) do
4:     if enabled\( (t) \land KB \models t.\phi \) then
5:       handleTransition\( (t) \)
6:     CurrentMarking = fire\( (t) \)
7:   end if
8: end for
9: end while
```

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)
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 4-Legged Striker

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)

Petri Net Plans
Example: A Simple 4Legged Striker

```
ex.approachBall

end.approachBall

ex.seekBall
end.seekBall
start.seekball

start.approachBall
ex.approachBall
end.approachBall
haveBall

start.trackBall
ex.trackBall
(not seenBall)

ex.seekBall
end.seekBall
seenBall

start.approachBall

end.approachBall
haveBall

Goal
```
Example: A Simple 4Legged Striker

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)
Example: A Simple 4Legged Striker

Petri Net Plans
Example: A Simple 4Legged Striker

```
start.approachBall
ex.approachBall
end.approachBall.haveBall
start.trackBall
ex.trackBall
(ex.not seenBall)
end.trackBall
Goal
```
Example: A Simple 4Legged Striker

```
ex.seekBall
end.seekBall.seenBall
start.seekBall
start.approachBall
ex.approachBall
end.approachBall.haveBall
start.trackBall
end.trackBall
Goal
(ex.trackBall)
```

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)
Example: A Simple 4Legged Striker

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)

Petri Net Plans
Example: A Simple 4Legged Striker

V. A. Ziparo, L. Iocchi, D. Nardi ( RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma )
Example: A Simple 4Legged Striker

- **start**.approachBall
- **ex**.approachBall
- **end**.approachBall.haveBall
- **ex**.seekBall
- **end**.seekBall.seenBall
- **start**.seekball
- **ex**.trackBall
- **end**.trackBall

Goal

V. A. Ziparo, L. Iocchi, D. Nardi ( RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)
Example: A Simple 4Legged Striker

Goal

Petri Net Plans
Example: A Simple 4-Legged Striker

- **start.seekball**
- **end.seekBall.seenBall**
- **start.approachBall**
- **ex.approachBall**
- **end.approachBall.haveBall**
- **start.trackBall**
- **end.trackBall**
- **ex.trackBall**

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma)
Example: A Simple 4Legged Striker

Basic PNPs

Ex. seekBall
End. seekBall. seenBall
Start. seekBall
Start. approachBall
End. approachBall. haveBall
Start. trackBall
End. trackBall
Goal

(V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma))
Example: A Simple 4Legged Striker

```
start.seekBall

ex.seekBall

end.seekBall.seenBall

start.approachBall

ex.approachBall

end.approachBall.haveBall

start.trackBall

ex.trackBall

(not seenBall)

end.trackBall

Goal
```
Example: A Simple 4Legged Striker

```
ex.seekBall
end.seekBall.seenBall start.seekball
start.approachBall
ex.approachBall
start.trackBall
end.trackBall
Goal
ex.trackBall
(not seenBall)
end.approachBall.haveBall
```

V. A. Ziparo, L. Iocchi, D. Nardi ( RoCoCo Laboratory Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti" "Sapienza" Università di Roma )
Example: A Simple 4Legged Striker

- `start.seekBall`
- `ex.seekBall`
- `end.seekBall`
- `seenBall`
- `start.approachBall`
- `ex.approachBall`
- `end.approachBall`
- `haveBall`
- `start.trackBall`
- `ex.trackBall`
- `(not seenBall)`

Petri Net Plans
**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 ($K_{B_i}$).
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 $\{M_0, \ldots, M_n\}$, 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]
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

\[ p_{i1} \xrightarrow{t_f} p_{o1} \]
\[ p_{i2} \xrightarrow{t_j} p_{o2} \]

V. A. Ziparo, L. Iocchi, D. Nardi (RoCoCo Lab)
Soft Synchronization Decomposition

(a)
Multi-Robot Interrupt

![Petri Net Diagram]
Multi-Robot Interrupt Decomposition

(a) R1.rotate

int.(R1.lostBall)

R2.prepareToReceive

(b) rotate

R1.lostBall

prepareToReceive
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

1. Introduction
2. Petri Nets in a Nutshell
3. Basic PNPs
4. Multi-Robot
5. Conclusions
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.
Any Questions?
D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi.
Multi-objective exploration and search for autonomous rescue robots.

H. Costelha and P. Lima.
Modelling, analysis and execution of robotic tasks using petri nets.

Edmund H. Durfee.
Distributed problem solving and planning.

Assignment of dynamically perceived tasks by token passing in multi-robot systems.

ISSN:0018-9219.

L. Iocchi, D. Nardi, L. Marchetti, and V. A. Ziparo.
S.P.Q.R. Legged Team Description Paper.

K. Konolige.
COLBERT: A language for reactive control in saphira.

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
Conclusions


A robotic soccer passing task using Petri Net Plans (demo paper).

Weihua Sheng and Qingyan Yang.
Peer-to-peer multi-robot coordination algorithms: petri net based analysis and design.

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

Bericht 272, FBI-HH-B-272/06.