IM_FMT@IFM'2009, 16 February 2009

Semantic Embedding of Petri-Nets into Event-B

Christian Attiogbé
LINA - UMR 6241 University of Nantes

Outline

- Motivations
- B Abstract Systems and Modelling
- Petri Nets and Modelling
- The proposed solution
- Analysis of the Embedding of B and P-net
- Prototype Tool
- Summary and Future Work

General motivations (1/2)

Large software systems analysis and design are difficult engineering tasks

- Concepts, languages, tools, methods
- Composition of several (heterogeneous) components
  Asynchronous systems
- Complexity of asynchronous models

Developers need :

- Practical Guidelines
- Reliable models
- Rigorous analysis methods and techniques (involving several tools)

Complexity of systems $\Longrightarrow$

- Integration/Combination of appropriated methods and techniques

General motivations (2/2)

Combinaison: graphical structuration, Specifications extraction, Composition, Refinement, Model checking, Theorem proving

- Petri Nets : graphical, simulation and model verification frameworks
- B Method : refinement based approach, theorem proving

Complementarity

- graph exploration, *liveness*
- composition, refinement, *safety*

Outline

- Motivations
- B Abstract Systems and Modelling
- Petri Nets and Modelling
- The proposed solution
- Analysis of the Embedding of B and P-net
- Prototype Tool
- Summary and Future Work

Event-B Specification Approach

Abstract systems are used to structure Event-B specifications
An abstract system is a mathematical model of an asynchronous system behaviour

An abstract system is made of:

- state space description: invariant (constants, sets, variables, ...),
- Event descriptions; e $\widehat{=}$ eGuard $\Longrightarrow$ eBody
  They are guarded actions/substitutions.

Refinement of abstract systems (data and events)
Decomposition of abstract systems (of a system into sub-systems)

Events

An event has one of the following general forms:

(SELECT Form)	(ANY Form)
`name` $\widehat{=}$ /* event name */ select $P(gcv)$ then $GS(gcv)$ end	`name` $\widehat{=}$ /* event name */ any bv where $P(bv, gcv)$ then $GS(bv, gcv)$ end

Figure: General Forms of Events

bv denotes the local bound variables of the event;

gcv denotes the global constants and variables of the abstract;

$P(bv, gcv)$ a predicate.

Abstact System : Semantics and Consistency

- the initialisation U establishes the invariant ($I(gcv)$);

$$[U]I(gcv)$$

- each event preserves the invariant :
  In the case of an event with the ANY form, the proof obligation is:

$$I(gcv) \wedge P(bv, gcv) \wedge \mathsf{prd}_v(S) \Rightarrow [GS(bv, gcv)]I(gcv)$$

- The events (e) terminate:

$$I(gcv) \wedge eGuard \Rightarrow \mathsf{fis}(eBody)$$

- There is always at least one guard true

$$eGuard_1 \vee eGuard_1 \vee \cdots$$

Abstact System : Semantics and Consistency

The predicate fis(S) expresses that S does not establish $False$:

$$\text{fis}(S) \Leftrightarrow \neg\, [S] False$$

ie

$$I(gcv) \wedge eGuard \Rightarrow \neg\, [S] False$$

The predicate $\text{prd}_v(S)$ is the *before-after predicate* of the substitution S ; it relates the values of state variables just before (v) and just after (v') the substitution S.

The prd_v(any x where $P(x, v)$ then $v := S(x, v)$ end) is:

$$\exists\, x.(P(x, v) \wedge v' = S(x, v))$$

Refinement

Data refinement (as usually)
Event Refinement:

- Introduction of new events.
- Strengthening guards (unlike with Classical B)
- Each event of the concrete system refines an event of the abstraction.

Let A with Invariant: I(av)

```
evt_a ≙ /* Abs. ev. */
   when P(av)
   then GS(av)
   end

prd_v(...) = Ba(av, av')
```

Refined with: Invariant J(av,cv)

```
evt_r ≙ /* Conc. ev. */
   when Q(cv)
   then GS(cv)
   end

prd_v(...) = Bc(cv,cv')
```

PO: $I(av) \wedge J(av, cv) \wedge Q(cv) \wedge Bc(cv, cv') \Rightarrow \exists\, cv'.(Ba(av, av') \wedge J(av', cv'))$

Outline

- Motivations
- B Abstract Systems and Modelling
- Petri Nets and Modelling
- The proposed solution
- Analysis of the Embedding of B and P-net
- Prototype Tool
- Summary and Future Work

Petri Nets

A Petri Net (P-net) is a 4-tuple $(P, T, Pre, Post)$:

- P a finite set of places
(with $\mid P \mid = m$, the cardinality of P)
- T a finite set of transitions,
(with $\mid T \mid = n$, the cardinality of T)
- $Pre : P \times T \rightarrow \mathbb{N}$ a mapping (an m x n array),
- $Post : P \times T \rightarrow \mathbb{N}$ a mapping (an m x n array).

Petri Nets

A Petri Net (P-net) is a 4-tuple $(P, T, Pre, Post)$:

- P a finite set of places
  (with $\mid P \mid = m$, the cardinality of P)
- T a finite set of transitions,
  (with $\mid T \mid = n$, the cardinality of T)
- $Pre : P \times T \rightarrow \mathbb{N}$ a mapping (an m x n array),
- $Post : P \times T \rightarrow \mathbb{N}$ a mapping (an m x n array).

A marked net is a couple $M = (N, \mu)$ made of a net N and a map $\mu : P \rightarrow \mathbb{N}$.
$\mu(p)$ is the marking of the place p, the number of tokens (or marks) within p

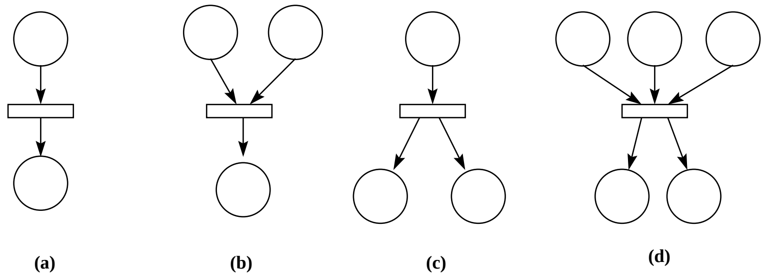
Graph associated to a P-net.

The graph associated to a net N is:

- Γ_p the transitions reachable from each place:
  $\forall\, p \in P \,.\, \Gamma_p(p) = \{t \in T \mid Pre(p,\, t) > 0\}$
- Γ_t the places reachable from each transition:
  $\forall\, t \in T \,.\, \Gamma_t(t) = \{p \in P \mid Post(p,\, t) > 0\}$
- W_{in} the weight of each input edge:
  $\forall\ p \in P, \forall\ t \in T \,.\, W_{in}(p,\, t) = Pre(p, t)$ and
- W_{out} the weight of each output edge:
  $\forall\ p \in P, \forall\ t \in T \,.\, W_{out}(p,\, t) = Post(p, t)$

The graph associated to a P-net is the abstract representation which is used to manipulate the net.

P-net: behaviour, evolution



Net evolution: firing of transitions

Evolution of the net and marking → Graph

Outline

- Motivations
- B Abstract Systems and Modelling
- Petri Nets and Modelling
- The proposed solution
- Analysis of the Embedding of B and P-net
- Prototype Tool
- Summary and Future Work

Integration technique: embedding

- Shallow *Embedding*: a given specification in a language is translated into a corresponding specification in another language.
- Semantic *Embedding*: a given logic/language and its semantics are translated into another logic/language

From Petri-Net to Event-B

- Formal analysis in B of systems described with P-net :
  - Proof of safety properties, refinement,
  - Tool Integration in the same software project

$\Rightarrow$

- Semantic Embedding of Petri nets into Event-B

$\Rightarrow$

- Find a generic B structure for P-net
  Systematic translation ✍ tools: PN2B

P-net: structure

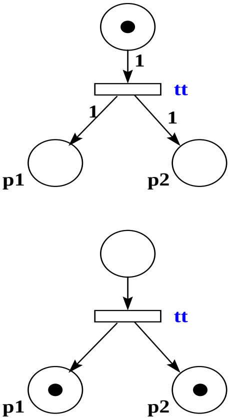
Construction of a graph

- Sets of places and transitions
- Relations between places and transitions
- Marking Functions: places, transitions
- invariant Properties

⇒ an abstract B system (State space description)

it remains the evolution of the net based on transitions and markings.

Capturing the P-net evolution semantics



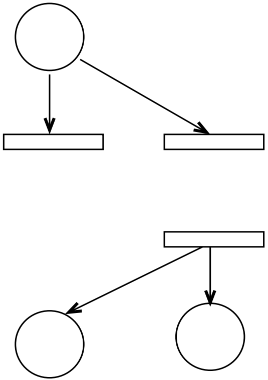
tt enabled (firable):

- Guard: input places well marked
- Effect :
  update of input places and update of output places

→ instantaneous!

the pi may fire other transitions,...

Abstraction, modelling



$$\Gamma_b : Transitions \leftrightarrow Places$$

(placesBefore)

$$\Gamma_a : Transitions \leftrightarrow Places$$

(placesAfter)

Abstraction, modelling in B

A P-net N is a 4-tuple $(Places, Transitions, placesAfter, placesBefore)$ where:

- $placesAfter : Transitions \leftrightarrow Places$
- $placesBefore : Transitions \leftrightarrow Places$

A marked net is a couple $M = (N, \mu)$

- $\mu : Places \rightarrow \mathbb{N}$ (total function)

the weight of the transitions may be ≥ 1:

$$weightBefore : Transitions \times Places \nrightarrow \mathbb{N}$$

$$\mathrm{dom}(weightBefore) = placesBefore$$

Abstraction, modelling in B

A transition t is enabled (firable) if

$$\forall\, p \in placesBefore(t).\, \mu(p) \geq 1$$

If a transition is fired:

- $\forall\, p \in placesBefore(t).\, \mu(p) := \mu(p) - 1$
- $\forall\, p \in placesAfter(t).\, \mu(p) := \mu(p) + 1$

Generalisation :
$weightBefore(t,p)$ and $weightAfter(t,p)$ in place of 1

Abstraction, modelling in B

A transition is an event of the abstract system.
The guard of the event is a right marking of the input places (enabling)

```
event_ti ≙                /* any ti transition */
   any ti, ··· where
      firable(ti, ···) /* the idea to be refined */
   then
      Substitution to update of the markings wrt ti
   end
```

Abstraction, modelling in B

```
event_tr =
    ANY ti ... WHERE
        ∀ p ∈ placesBefore(ti). μ(p) ≥ 1
    THEN
        /* ∀ p ∈ placesBefore(ti). μ(p) := μ(p) − 1 */
        /* ∀ p ∈ placesAfter(it). μ(p) := μ(p) + 1 */
    END
```

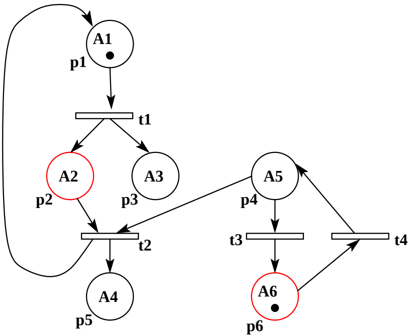
Rough idea to be correctly modelled in B

Abstraction, modelling in B

In the case of High Level Petri Nets (HLPN)

- Actions/Operations performed in the places
- They need 'time'
- The places and the actions are guarded by the transitions.
- Firing the transitions in 2 steps:
  - to enable the guards
  - to launch the actions until completion

Abstraction, modelling in B



- t1 enables { A2, A3}
- t2 enables { A4 }
- t3 enables { A6 }
- t4 enables { A5 }

The ti impact on guard(Ai)

guard(Ai) := True

$\forall\ Ai \in placesAfter(ti)$

Abstraction, modelling in B

Step 1:

- Each action (in the places) of a P-net is a guarded event
- The guard of an action is enabled if the related transition is fired
- Several places guarded by a transition $\Rightarrow$ several enabled actions

Abstraction, modelling in B

- nondeterministic choice, but all actions should be performed
- A guard stay True until the action is performed
- After the running of an action, its guard becomes false; update of marks

- Hypothesis: atomicity of actions
- Practically : to wait the end of the action

Evolution abstraction, modelling in B

A transition

- whose before places are conveniently marked
- is enabled (firable)
- and enables the guards of all the actions (in before places)

Abstraction: Evolution semantics (1)

event_tr = ANY t ... becomes

fire_transition =

ANY t ... WHERE

$\forall p \in placesBefore(t).\, \mu(p) \geq 1$

& ... & tmp_guarded_acts = ran(involved_actions)*TRUE

THEN

- $\forall p \in placesBefore(t).\, \mu(p) := \mu(p) - 1$
- $\underbrace{\forall p \in placesAfter(t).\, \mu(p) := \mu(p) + 1}$ *replaced by*
- guarded_actions := guarded_actions $\lhd\!\!-$ tmp_guarded_acts

END

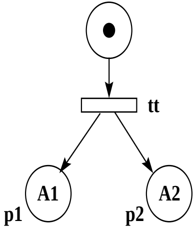
The previous rough idea is properly refined!

Abstraction: Evolution semantics (2)

Step 2:
After the firing of the transitions (step 1):

- launching of the actions (the corresponding events)
- All action with a guard set to True
- in a nondeterministic way

Formalisation in Event-B




```
Action_A1 =
ANY pp WHERE ...
& guarded_actions(A1) = true
& pp = p1
THEN
/* A1 */
mu(pp) := mu(pp) + weightAfter(tt,
|| guarded_action(A1) := false
END
;
Action_A2 = ... /* idem */
```

Formalisation in Event-B

```
action_Ak = /* any action Ak (in a place) */
ANY pp WHERE /* pp = place of the action Ak */
pp : PLACE & pp = treatement (Ak)
& guarded_actions(Ak) = true
THEN
/* the action Ak */
guard(Ak) := false
|| mu(pp) := mu(pp) + ...
END
```

Synthesis of the solution: Embedding P-net into Event-B

- Semantic of Petri net ?

  structure + evolution semantics (Basic P-net or HLPN)

✍

- A B abstract system Pnet capturing the structure of a P-net
- A B abstract system EmbeddedPN[Pnet] capturing the semantics

Formalisation of P-net in Event-B

```
MACHINE
PetriNet /* structure */
SETS
PLACE
; TRANSITION
; ACTION = {aj, ak, nullac-
tion}
; NET_Type = {pure, unspeci-
fied, colored}
; NET_Mode = {edition, anal-
ysis}
```

```
VARIABLES
places
, transitions
, placesBefore
, placesAfter
, mu /* marking of each place */
, weightBefore
, weightAfter
, net_type /* PN type */
, net_mode /* PN mode */
, actions /* attached to pl. */
, treatment /* associated to pl.
*/
```

Formalisation of P-net in Event-B

```
INVARIANT
places <: PLACE
& transitions <: TRANSITION
& placesBefore : transitions <-> places
& placesAfter : transitions <-> places
& mu : places -> NAT
& weightBefore : transitions * places +-> NAT
& dom(weightBefore) = placesBefore
& weightAfter : transitions * places +-> NAT
& dom(weightAfter) = placesAfter
...
```

Formalisation of P-net in Event-B

```
INVARIANT (continued)
& dom(placesBefore) = transitions
/* every transition has at least one place after it */
& dom(placesAfter) = transitions
& net_type : NET_Type
/* no cycle : place_i -> trans-i -> place_i */
& ((net_type = pure) => (placesBefore / placesAfter = ))
& net_mode : NET_Mode
& actions <: ACTION /* all the actions controlled by the PN */
& nullaction : actions
& treatment : places -> actions

/* now we add some general SAFETY properties */
/* mode = analysis ==> the properties */
/* complete the shape ((mode = analysis) => ()) */
```

Formalisation of P-net in Event-B

```
INITIALISATION
...
OPERATIONS
add_transition = /* add a transition between two existing places */
;
add_place = /* add a place to the P Net */
;
add_place_before = /* add an existing place pl before an existing t
;
add_place_after =/* add an existing place pl after an existing tr *
;
modify_mark = /* set the token in a place pp to vv */
;
...
```

Formalisation of P-net in Event-B

```
...
modify_beforeEdge_weight = /* modifiy the weight of the edge
before a tr */
;
modify_afterEdge_weight = /* modifiy the wight of the edge af-
ter a tr */
;
set_analysis_mode = /* set the mode to analysis */
;
set_edition_mode = /* set the mode to edition */
;
res <- which_mode = /* what is the current mode */
;
set_mu = /* set mu to the prm nmu */

END
```

Specification structuring

```
MACHINE
EmbeddedPN /* EmbeddedPN[Pnet]
*/
INCLUDES
PetriNet
VARIABLES
...
INVARIANT
...
INITIALISATION
...
```

```
OPERATIONS
fire_transition = ...
;
action_Ai = ...
;
action_Aj = ...
;
...

END
```

Outline

- Motivations
- B Abstract Systems and Modelling
- Petri Nets and Modelling
- The proposed solution
- Analysis of the Embedding of B and P-net
- Prototype Tool
- Summary and Future Work

Screenshot of the prototype tool

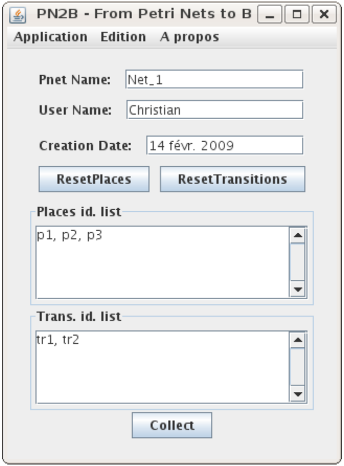


Figure: Main window of the GUI

Screenshot of the prototype tool

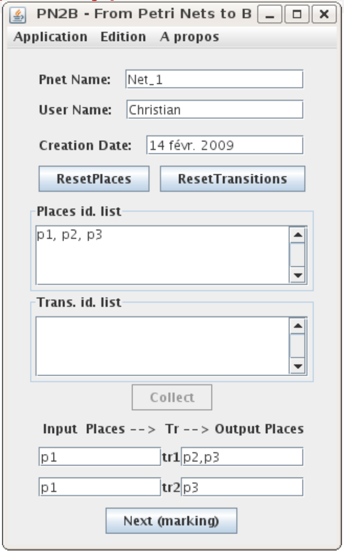
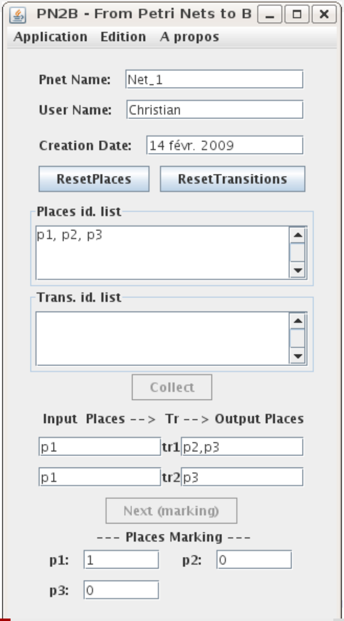


Figure: GUI - Description of the net

Screenshot of the prototype tool



The generated B specification

```
/*-------------------
B Model generated by the PN2B Modeler
From COLOSS @ LINA (Nantes University)
-------------------
Generated for Christian Date : 14 févr. 2009
--------------------*/
MACHINE Net_1
SETS
PLACES=p1, p2, p3
; TRANSITIONS=tr1, tr2
; ACTION = aj, ak, tai, taj, nullaction /* actions associated */
; NET_Type = pure, unspecified, colored
; NET_Mode = edition, analysis /* edition or analysis mode */
CONSTANTS /* parameter of the machine */
places /* the places in the PN */
, transitions /* the transitions in the PN */
, placesBefore /* places before a transition */
, placesAfter /* places after a transition */
, weightBefore /* weight of an edge before a transition */
, weightAfter /* weight of an edge after a transition */
```

The generated B specification

```
PROPERTIES
places <: PLACE
& transitions <: TRANSITION
& placesBefore : transitions <-> places
& placesAfter : transitions <-> places
& weightBefore: transitions * places +-> NAT
& dom(weightBefore) = placesBefore & weightAfter : transitions *
places +-> NAT
& dom(weightAfter) <: placesAfter
& dom(placesBefore) <: transitions
/*every transition has at least one place after it */
& places = p1, p2, p3
& transitions = tr1, tr2
& PlacesBefore =  tr1 |-> p1, tr2 |-> p1
& PlacesAfter =  tr1 |-> p2, tr1 |-> p3, tr2 |-> p3
& dom(placesAfter) <: transition
& weightAfter = placesAfter*1 /* weight of edge after a transition */
& weightBefor = placesBefore*1
& pl_actions <: ACTION
/* the actions controlled by the PNet */
```

Outline

- Motivations
- B Abstract Systems and Modelling
- Petri Nets and Modelling
- The proposed solution
- Analysis of the Embedding of B and P-net
- Prototype Tool
- Summary and Future Work

Summary and conclusion

Methods and Tools Integration

- B tools behind P-net graphical user interface

Formal Analysis: proving properties

- Consistency proof
- Safety properties added to the invariant

Related approaches

- CSP to B (M. Butler)
- Statecharts to B (Emil Sekerinski)

Summary and conclusion

A Semantic embedding of Petri nets into Event-B

- Design, simulation, exploration: Petri nets
- Composition, refinement (coding): B

✍ Bridging Event-B and P-nets Basic P-nets and HLPN
Prototype Tool

- Case studies, experimentations
- Multi-facet specification and analysis methods
- From Event-B to Petri nets?
- Shallow embedding for specific properties?

Thank you

Questions, please