Formal Proof of Petri Net Refinement using Coq

Christine Choppy   Micaela Mayero   Laure Petrucci

Laboratoire d'Informatique de l'Université Paris Nord (LIPN)
Université Paris 13

TYPES 2006

Motivations

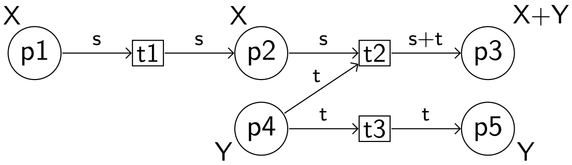
- Petri Nets are a formalism for modeling and validating critical systems: protocols, state systems, ...
- Refinement techniques to obtain a more concrete specification from an abstract view: to deal with the number of states.
- To prove formally and automatically the refinement relation between two nets: to keep interesting properties.
- Using formal proofs in the verification domain.
- First experiment.

Definition of Colored Petri Net

Tuple $N = (P, T, A, C, E, M, Y, M_0)$ where:

1. P is a set of places
2. T is a set of transitions
3. A is a set of arcs
4. C determines the colors of places and transitions
5. E gives the arc inscriptions
6. M is the set of markings
7. Y is the set of steps
8. M_0 is the initial marking

Example



Kind of refinement

3 kinds of refinement:

Kind of refinement

3 kinds of refinement:

- subnet refinement.

Kind of refinement

3 kinds of refinement:

- subnet refinement.
- node refinement.

Kind of refinement

3 kinds of refinement:

- subnet refinement.
- node refinement.
- type refinement (colors).

Subnet refinement

N_r is a refined subnet of N if:

Subnet refinement

N_r is a refined subnet of N if:

- $P_{N_r} \subset P_N$

Subnet refinement

N_r is a refined subnet of N if:

- $P_{N_r} \subset P_N$
- $T_{N_r} \subset T_N$

Subnet refinement

N_r is a refined subnet of N if:

- $P_{N_r} \subset P_N$
- $T_{N_r} \subset T_N$
- $A_{N_r} \subset A_N$

Subnet refinement

N_r is a refined subnet of N if:

- $P_{N_r} \subset P_N$
- $T_{N_r} \subset T_N$
- $A_{N_r} \subset A_N$
- There is no new arc from places of N to transitions on N_r.

Subnet refinement

N_r is a refined subnet of N if:

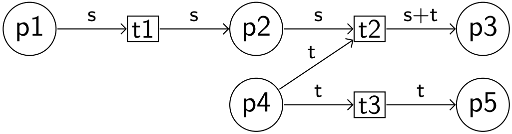
- $P_{N_r} \subset P_N$
- $T_{N_r} \subset T_N$
- $A_{N_r} \subset A_N$
- There is no new arc from places of N to transitions on N_r.
- There is no new arc from transitions of N to places of N_r.

Subnet refinement

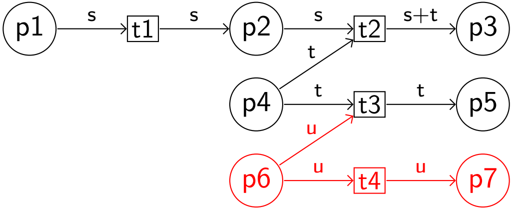
N_r is a refined subnet of N if:

- $P_{N_r} \subset P_N$
- $T_{N_r} \subset T_N$
- $A_{N_r} \subset A_N$
- There is no new arc from places of N to transitions on N_r.
- There is no new arc from transitions of N to places of N_r.
- There is no new arc from transitions of N_r to places of N.

Subnet refinement example



Subnet refinement example

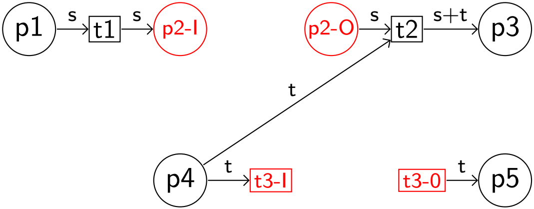


Node refinement

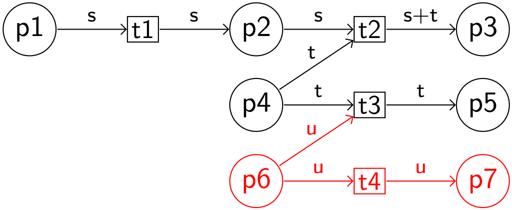
N_r is a refined subnet of N if the number of tokens are preserved in I/O:

Node refinement

N_r is a refined subnet of N if the number of tokens are preserved in I/O:



Subnet refinement



Coq formalization (initial net)

```
Require Export Ensembles Constructive_sets
               Constructive_sets Finite_sets_facts.
```

Coq formalization (initial net)

```
Require Export Ensembles Constructive_sets
               Constructive_sets Finite_sets_facts.

Inductive places :  Type := p :  nat -> places.
```

Coq formalization (initial net)

```
Require Export Ensembles Constructive_sets
               Constructive_sets Finite_sets_facts.

Inductive places :  Type := p :  nat -> places.
Definition arc_pt (pl:places)(tr:transitions)(n:nat):=
       (pairT (pairT pl tr) n).
```

Coq formalization (initial net)

```
Require Export Ensembles Constructive_sets
               Constructive_sets Finite_sets_facts.

Inductive places :  Type := p :  nat -> places.
Definition arc_pt (pl:places)(tr:transitions)(n:nat):=
       (pairT (pairT pl tr) n).

Definition Pi1 := p 1.
```

Coq formalization (initial net)

```
Require Export Ensembles Constructive_sets
               Constructive_sets Finite_sets_facts.

Inductive places :  Type := p :  nat -> places.
Definition arc_pt (pl:places)(tr:transitions)(n:nat):=
       (pairT (pairT pl tr) n).

Definition Pi1 := p 1.

Definition Pi :=
      Add places (Add places (Add places (Add places
      (Add places (Empty_set places) Pi1) Pi2)
      Pi3) Pi4) Pi5.
```

Coq formalization (initial net)

```
Require Export Ensembles Constructive_sets
               Constructive_sets Finite_sets_facts.

Inductive places :  Type := p :  nat -> places.
Definition arc_pt (pl:places)(tr:transitions)(n:nat):=
       (pairT (pairT pl tr) n).

Definition Pi1 := p 1.

Definition Pi :=
      Add places (Add places (Add places (Add places
      (Add places (Empty_set places) Pi1) Pi2)
      Pi3) Pi4) Pi5.

Definition Aip1t1 := arc_pt Pi1 Ti1 1.
Definition Ait2p3 := arc_tp Ti2 Pi3 2.
```

Coq formalization (refined net)

```
Definition P :=
  Add places (Add places (Add places (Add places (Add places
  (Add places (Add places  (Empty_set places)
     P1) P2) P3) P4) P5) P6) P7.

Definition T :=
  Add transitions (Add transitions (Add transitions
    (Add transitions (Empty_set transitions)
       T1) T2) T3) T4.
```

Subnet refinement

N_r is a refined subnet of N if:

- $P_{N_r} \subset P_N$
- $T_{N_r} \subset T_N$
- $A_{N_r} \subset A_N$
- There is no new arc from places of N to transitions on N_r.
- There is no new arc from transitions of N to places of N_r.
- There is no new arc from transitions of N_r to places of N.

Using subnet refinement properties

```
Lemma subnet_refined:

      Included places Pi P /\
      Included transitions Ti T /\
 (Included  (prodT (prodT places transitions) nat) Aipt Apt /\

   (forall (pe:places) (te:transitions) (n:nat),
      (In places Pi pe) -> (In transitions T te) ->
      ~In (prodT (prodT places transitions) nat)
   (Setminus (prodT (prodT places transitions) nat) Aipt Apt)
  (pairT (pairT pe te) n))) /\
```

similar for Aitp Atp...

Manual proof and automation

Simple: `unfold, auto with set,...`

```
Ltac is_subnet_refinement :=
  try
   match goal with
   | |- ?X1 /\ ?X2 =>
     split;[try (match goal with | |- (Included _ ?X3 ?X4) =>
          unfold X3, X4; auto with sets end) | intros;
     (match goal with | |- ~(In _ (Setminus _ ?X3 ?X4) _ )=>
          unfold X3, X4 end);unfold Setminus;intro; ... ]
   end.
```

Conclusion

- Case study.

Conclusion

- Case study.
- Subnet refinement achieved (including automation).

Conclusion

- Case study.
- Subnet refinement achieved (including automation).
- Node refinement almost achieved.

Conclusion

- Case study.
- Subnet refinement achieved (including automation).
- Node refinement almost achieved.
- Type refinement much more complicated (colors = properties).

Conclusion

- Case study.
- Subnet refinement achieved (including automation).
- Node refinement almost achieved.
- Type refinement much more complicated (colors = properties).
- First experiment using formal proofs in the Petri Net domain.

Future work (1)

Future work (1)

- To finish other refinement proofs for this case study.

Future work (1)

- To finish other refinement proofs for this case study.
- To develop an interface with Petri Net tools.
  - Manual definition for big Petri Nets is impossible.
  - Idea: to use FAST, PNML (XML for PN), ...

Future work (1)

- To finish other refinement proofs for this case study.
- To develop an interface with Petri Net tools.
  - Manual definition for big Petri Nets is impossible.
  - Idea: to use FAST, PNML (XML for PN), ...
- Full automation.

Future work (1)

- To finish other refinement proofs for this case study.
- To develop an interface with Petri Net tools.
  - Manual definition for big Petri Nets is impossible.
  - Idea: to use FAST, PNML (XML for PN), ...
- Full automation.
- Real case study.

Future work (2): missile simulator