3
Propositional Dynamic Logic for Petri Nets (Petri-PDL)

This chapter presents a Propositional Dynamic Logic that uses Petri Nets terms as programs (Petri-PDL). We define an axiomatic system and prove soundness, completeness, and the EXPTime-hardness of its satisfiability problem.

As pointed out by the work of Mazurkiewicz (1987, 1989), logics that deal with Petri Nets use to be incomplete due to the possibility of a place always increase its token amount (up to countable infinity). To restrict a subset of Petri Nets where we can achieve decidability and completeness, we call normalised Petri Net any Petri Net composed as in Section 2.3.2 and do not contains any place that can accumulate an infinity amount of tokens. From now on, all the proofs deals only with normalised Petri Nets.

3.1
Language and semantics

The language of Petri-PDL consists of

Propositional symbols: p, q..., where Φ is the set of all propositional symbols

Place names: e.g.: $a, b, c, d \ldots$

Transition names: e.g.: $t_1, t_2, t_3 \ldots$

Petri Net Composition symbol: $\odot$

Sequence of names: $S = \{\epsilon, s_1, s_2, \ldots\}$, where ϵ is the empty sequence. We use the notation $s \prec s'$ to denote that all names occurring in s also occur in s', regardless its order.

Definition 23 Petri-PDL program

We use π to denote a Petri Net program and s to denote a sequence of names (the markup of π). The transitions may be from three types, $T_1 : xt_1y$, $T_2 : xyt_2z$ and $T_3 : xt_3yz$, each transition has a unique type.

Basic programs: $\pi_b ::= at_1b \mid at_2bc \mid abt_3c$ *where t_i is of type $T_i, i = 1, 2, 3$*

Petri Net Programs: $\pi ::= s, \pi_b \mid \pi \odot \pi$

Definition 24 Petri-PDL formula

A formula is defined as

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\pi\rangle\varphi.$$

We use the standard abbreviations $\bot \equiv \neg\top$, $\varphi \vee \phi \equiv \neg(\neg\varphi \wedge \neg\phi)$, $\varphi \rightarrow \phi \equiv \neg(\varphi \wedge \neg\phi)$ and $[s,\pi]\varphi \equiv \neg\langle s,\pi\rangle\neg\varphi$.

Definition 25 Firing function

We define the firing *function* $f : S \times \pi \rightarrow S$ *as follows*

- $f(s, at_1b) = \begin{cases} s_1bs_2 & \text{if } s = s_1as_2 \\ \epsilon & \text{if } a \not\prec s \end{cases}$
- $f(s, abt_2c) = \begin{cases} s_1cs_2s_3 & \text{if } s = s_1as_2bs_3 \\ \epsilon & \text{if } a \not\prec s \text{ or } b \not\prec s \end{cases}$
- $f(s, at_3bc) = \begin{cases} s_1s_2bc & \text{if } s = s_1as_2 \\ \epsilon & \text{if } a \not\prec s \end{cases}$
- $f(\epsilon, \eta) = \epsilon$, *for all petri nets programs* π.
- $f(s,\pi) = \begin{cases} f(s,\eta) & \text{if } \exists\eta \subset \pi \text{ such that } \eta \text{ is a basic transition and } f(s,\eta) \neq \epsilon \\ \epsilon & \text{otherwise} \end{cases}$

Definition 26 Petri-PDL frame

A frame for Petri-PDL is a 3-tuple $\mathcal{F} = \langle W, R_\pi, M\rangle$, *where*

- W *is a non-empty set of states;*
- $M\colon W \rightarrow S$;
- R_π *is a binary relation over* W, *for each basic program* π, *satisfying the following condition. Let* $s = M(w)$
  - *if* $f(s,\pi) \neq \epsilon$, $wR_\pi v$ *iff* $f(s,\pi) \prec M(v)$
  - *if* $f(s,\pi) = \epsilon$, $wR_\pi v$ *iff* $w = v$

Definition 27 Behaviour of R over composed Petri Net programs

We inductively define the behaviour of the relation R_π, for each Petri Net program
$\pi = \pi_1 \odot \pi_2 \odot \cdots \odot \pi_n$, *as*

$$R_\pi = \{(w, v) \mid \text{ for some } \pi_i, \exists u \text{ such that } s_i \prec M(u) \text{ and } wR_{\pi_i}u \text{ and } uR_\pi v\}$$

Where $s_i = f(s, \pi_i)$, for all $1 \leq i \leq n$ and $s = M(w)$.

Definition 28 Petri-PDL model

A model *for Petri-PDL is a pair $\mathcal{M} = \langle \mathcal{F}, \mathbf{V} \rangle$, where $\mathcal{F}$ is a Petri-PDL frame and $\mathbf{V}$ is a valuation function $\mathbf{V} \colon \Phi \to 2^W$.*

The semantical notion of satisfaction for Petri-PDL is defined as follows.

Definition 29 Petri-PDL satisfaction notion

Let $\mathcal{M} = \langle \mathcal{F}, \mathbf{V} \rangle$ be a model. The notion of satisfaction *of a formula φ in a model $\mathcal{M}$ at a state w, notation $\mathcal{M}, w \Vdash \varphi$, can be inductively defined as follows:*

- $\mathcal{M}$,$w \Vdash p$ *iff* $w \in \mathbf{V}(p)$;
- $\mathcal{M}$,$w \Vdash \top$ *always;*
- $\mathcal{M}$,$w \Vdash \neg\varphi$ *iff* $\mathcal{M}$,$w \nVdash \varphi$;
- $\mathcal{M}$,$w \Vdash \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}$,$w \Vdash \varphi_1$ *and* $\mathcal{M}$,$w \Vdash \varphi_2$;
- $\mathcal{M}$,$w \Vdash \langle s, \eta \rangle \varphi$ *iff there exists* $v \in W$, $wR_\eta v$, $s \prec M(w)$ *and* $\mathcal{M}$,$v \Vdash \varphi$.

If $\mathcal{M}$,$v \Vdash A$ for every state v, we say that A is *valid in the model* $\mathcal{M}$, notation $\mathcal{M} \Vdash A$. And if A is valid in all $\mathcal{M}$ we say that A is *valid*, notation $\Vdash A$.

3.2 Axiomatic system

We consider the following set of axioms and rules, where p and q are proposition symbols, φ and ψ are formulae, $\eta = \eta_1 \odot \eta_2 \odot \cdots \odot \eta_n$ is a Petri Net program and π is a Petri Net program with a sequence of names.

(PL) All propositional logic tautologies

(K) $[s,\pi](p \to q) \to ([s,\pi]p \to [s,\pi]q)$

(Du) $[s,\pi]p \leftrightarrow \neg\langle s,\pi\rangle\neg p$

(PC) $\langle s,\eta\rangle\varphi \leftrightarrow \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$, where $s_i = f(s,\eta_i)$, for all $1 \leq i \leq n$ and π is not a basic program

($\mathbf{R}_\epsilon$) $\langle s,\eta\rangle\varphi \leftrightarrow \varphi$, if $f(s,\eta) = \epsilon$

(Sub) If $\Vdash \varphi$, then $\Vdash \varphi^\sigma$, where σ uniformly substitutes proposition symbols by arbitrary formulae

(MP) If $\Vdash \varphi$ and $\Vdash \varphi \to \psi$, then $\Vdash \psi$

(Gen) If $\Vdash \varphi$, then $\Vdash [s,\pi]\varphi$

Notice that the restriction over (PC) to not be applied over basic programs avoids the needing of axioms for basic programs.

3.3 Soundness and completeness

The axioms **(PL)**, **(K)** and **(Du)** and the rules **(Sub)**, **(MP)** and **(Gen)** are standard in the modal logic literature.

Lemma 30 *Validity of Petri-PDL axioms*

1. $\Vdash$ **PC**

   Proof: Suppose that there is a world w from a model $\mathcal{M} = \langle W, R_\pi, \mathbf{V}, M\rangle$ where PC is false. For PC to be false in w, there are two cases:

   (a) *Suppose $\mathcal{M}, w \Vdash \langle s,\eta\rangle\varphi$ (1) and $\mathcal{M}, w \nVdash \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$ (2); From Definition 29, (1) iff there is a v such that $wR_\eta v$, $s \prec M(w)$ and $\mathcal{M}, v \Vdash \varphi$ (3).*

By Definition 26 $R_\eta = \{(w,v) \mid$ *for some* $\eta_i, \exists u$ *such that* $s_i \prec M(u)$ *and* $wR_{\eta_i}u$ *and* $uR_\eta v\}$,
from (3) $\mathcal{M}, u \Vdash \langle s_i, \eta\rangle\varphi$ *and* $\mathcal{M}, w \Vdash \langle s, \eta_i\rangle\langle s_i, \eta\rangle\varphi$. *This implies* $\mathcal{M}, w \Vdash \langle s, \eta_1\rangle\langle s_1, \eta\rangle\varphi \vee \langle s, \eta_2\rangle\langle s_2, \eta\rangle\varphi \vee \cdots \vee \langle s, \eta_n\rangle\langle s_n, \eta\rangle\varphi$, *which contradicts (2).*

(b) *Suppose*
$\mathcal{M}, w \Vdash \langle s, \eta_1\rangle\langle s_1, \eta\rangle\varphi \vee \langle s, \eta_2\rangle\langle s_2, \eta\rangle\varphi \vee \cdots \vee \langle s, \eta_n\rangle\langle s_n, \eta\rangle\varphi$ *(2), by Definition 26, iff for some* i $(1 \leq i \leq n)$, $M, w \Vdash \langle s, \eta_i\rangle\langle s_i, \eta\rangle\varphi$ *iff there is a* u *such that* $wR_{\eta_i}u$, $s \prec M(w)$ *and* $\mathcal{M}, u \Vdash \langle s_i, \eta\rangle\varphi$ *(3), by Definition 29,*
iff there is a v *such that* $uR_\eta v$, $s_i \prec M(u)$ *and* $\mathcal{M}, v \Vdash \varphi$ *(4),*
By Definition 26, (3) and (4) we have $wR_\eta v$ *and* $s \prec M(w)$ *and* $M, v \Vdash \varphi$. *Thus,* $\mathcal{M}, w \Vdash \langle s, \eta\rangle\varphi$, *which contradicts the hypothesis.*

So, PC is valid. ■

2. $\Vdash \mathbf{R}_\epsilon$

Proof: Suppose that there is a world w *from a model* $\mathcal{M} = \langle W, R_\pi, M, \mathbf{V}\rangle$ *where* R_ϵ *is false. For* R_ϵ *to be false in* w, *there are two cases:*

(a) *Suppose* $\mathcal{M}, w \Vdash \langle \epsilon, \eta\rangle\varphi$ *(1) and*
$\mathcal{M}, w \nVdash \varphi$ *(2)*
(1) iff there is a v *such that* $wR_{\epsilon,\eta}v$ *and* $\mathcal{M}, v \Vdash \varphi$. *As* $f(\epsilon, \eta) = \epsilon$, *then* $w = v$ *and* $wR_\eta w$ *and* $\mathcal{M}, w \Vdash \varphi$, *which contradicts (2).*

(b) *Suppose* $\mathcal{M}, w \nVdash \langle \epsilon, \eta\rangle\varphi$ *(1) and*
$\mathcal{M}, w \Vdash \varphi$ *(2).*
(1) iff for all v *such that, if* $wR_{\epsilon,\eta}v$ *then* $\mathcal{M}, v \Vdash \neg\varphi$. *As* $f(\epsilon, \eta) = \epsilon$, *then* $w = v$ *and* $wR_\eta w$ *and* $\mathcal{M}, w \Vdash \neg\varphi$, *which contradicts (2).*

So, R_ϵ *is valid.* ■

Then, Petri-PDL is sound.

The completeness proof goes as in the work of Blackburn et al. (2001); Harel et al. (2000) and Goldblatt (1992*b*).

Definition 31 Filtration

Given a Petri-PDL formula φ, *a Petri-PDL model* $\mathcal{K} = \langle W, R_\eta, M, \mathbf{V}\rangle$, *we define a new model*

$$\mathcal{K} = \langle W^\varphi, R^\varphi_\eta, M^\varphi, \mathbf{V}^\varphi\rangle,$$

the filtration of $\mathcal{K}$ *by* $FL(\varphi)$*, as follows.*

The relation $\equiv$ *over the worlds of* $\mathcal{K}$ *is defined as*

$$u \equiv v \leftrightarrow \forall \phi \in FL(\varphi), \mathcal{K}, u \Vdash \phi \text{ iff } \mathcal{K}, v \Vdash \phi$$

(the equivalence class of u) and the relation R_η^φ *is defined as*

$$[u]R_\eta^\varphi[v] \leftrightarrow (\exists u' \in [u] \wedge \exists v' \in [v] \wedge u' R_\eta v').$$

(a) $[u] = \{v \mid v \equiv u\}$*, where* $[u]$ *is the equivalence class of* u

(b) $W^\varphi = \{[u] \mid u \in W\}$

(c) $[u] \in \mathbf{V}^\varphi(p)$ *iff* $u \in V(p)$

(d) $M^\varphi([u]) = \langle s_1, s_2, \dots \rangle$ *where for all* $j \geq 1, v_j \in [u]$ *iff* $M(v_j) = s_j$

All rules may be composed inductively to extend in order to compound all programs and propositions due to compositions as in section 3.1.

Lemma 32 *Filtration Lemma*

$\forall u, v \in W,\ uR_\eta v$ *iff* $[u]R_\eta^\varphi[v]$

Proof: As in Definition 31 $w \in [w]$ *iff* $\forall w' \in [w],\ w' \equiv w$ *(1)*
and $[u]R_\eta^\varphi[v]$ *for some* $u \in [u]$ *and* $v \in [v]$ *we have that* $u' R_\eta^\varphi v'$ *(2).*
So if $uR_\eta^\varphi v$ *and we do not have that* $[u]R_\eta^\varphi[v]$ *then it will contradicts (1). If* $[u]R_\eta^\varphi[v]$ *but we do not have that* $uR_\eta^\varphi v$ *then it will contradicts (2).* ■

Lemma 33 $\mathcal{K}^\varphi$ *is a finite Petri-PDL model.*
Proof:

- W^φ *is finite a set of states by Definition 31 and lemma 43.*
- $M^\varphi \colon W^\varphi \to S$ *by Definition 31.*
- $R_\eta^\varphi = \{([w], [v]) \mid$ for some η_i $\exists [u]$ such that $s_i \prec M^\varphi([u])$ and $[w]R_{\eta_i}[u]$ and $[u]R_\eta[v]\}$ *for any program* $\eta = \eta_1 \odot \cdots \odot \eta_n$*, where* $s_i = f(s, \eta_i)$ *and* $1 \leq i \leq n$*.*
- $\mathbf{V}^\varphi \colon \Phi \to 2^{W^\varphi}$ *by Definition 31.*

Then $\mathcal{K}^\varphi$ *is a finite Petri-PDL model.* ■

Corollary 34 *Decidability*
Proof: As, by Lemma 33, the number of states is finite, there is an algorithm to check whether a formula φ *of Petri-PDL is satisfiable.* ■

Definition 35 Canonic Model

The canonic model for Petri-PDL with language Λ *is a 4-tuple* $\mathcal{C}^\Lambda = \langle W^\Lambda, R^\Lambda, M^\Lambda, \mathbf{V}^\Lambda \rangle$*, where* W^Λ *is the set of all maximal consistent sets of formulae;* $\mathbf{V}^\Lambda$ *is the valuation function where for all* $w \in W^\Lambda$*,* $w \in \mathbf{V}^\Lambda(\varphi)$ *iff* $\varphi \in w$*;* M^Λ *is the markup of the Petri Net programs, defined as*

$$M^\Lambda(w) = \{s_1, \ldots, s_n \mid \langle s_i, \pi \rangle \varphi \in w, 1 \leq i \leq n, w \in W^\Lambda\};$$

and R^Λ *is the binary relation between the elements of* W^Λ *defined for each program* π *as*

$$R^\Lambda_\pi = \{(n, m) \mid n,m \in W^\Lambda, \{\varphi / [s, \pi]\varphi \in n, s \prec M^\Lambda(n)\} \subseteq m\}.$$

Lemma 36 $\mathcal{C}^\Lambda$ *is a model for Petri-PDL*
Proof: By Definition 35:

- W^Λ *is a set of states.*
- $M^\Lambda \colon W^\Lambda \to S$.
- $R^\Lambda_\pi = \{(w, v) \mid$ for some π_i $\exists u$ such that $s_i \prec M^\Lambda(u)$ and $wR_{\pi_i}u$ and $uR_\pi v\}$ *for any program* $\pi = \pi_1 \odot \cdots \odot \pi_n$*, where* $s_i = f(s, \pi_i)$ *and* $1 \leq i \leq n$.
- $\mathbf{V}^\Lambda \colon \Phi \to 2^{W^\Lambda}$.

So, $\mathcal{C}^\Lambda$ *is a model for Petri-PDL.* ■

Lemma 37 $[s, \pi]\varphi \in u$ *iff in all* v *such that* $uR^\Lambda_\pi v$*,* $\varphi \in v$.
Proof:

Supose $[s, \pi]\varphi \in u$ *and there is no* $v \in W^\Lambda$ *such that* $uR^\Lambda_\pi v$ *and* $\varphi \in v$ *(1).*

As $\pi = \pi_1 \odot \cdots \odot \pi_n$*, by the definition of* R^Λ_π *we have that all* $[s, \pi_i][s_i, \pi]\varphi, 1 \leq i \leq n \in v$ *for all* $1 \leq i \leq n$ *if* $uR^\Lambda_\pi v$ *(2).*

By (PC), all $[s, \pi_i][s_i, \pi]\varphi, 1 \leq i \leq n \in u$ *for all* $1 \leq i \leq n$ *(3).*

But if (3) then φ *is in some* v *such that* $uR^\Lambda_\pi v$ *by* R^Λ *definition, which contradicts (1).*

So, in all v *such that* $uR^\Lambda_\pi v$*,* $\varphi \in v$.

Supose that exists some $u \in W^\Lambda$ *where* $[s, \pi]\varphi \notin u$ *and such that in all* v *that* $uR^\Lambda_\pi v$*,* $\varphi \in v$ *(4).*

By R^Λ_π *definition, if in all* v *that* $uR^\Lambda_\pi v$*,* $\varphi \in v$*, then* $[s, \pi]\varphi \in u$ *(5).*

Then, there is a contradiction. ■

Lemma 38 *Let $\mathcal{C}^\Lambda = \langle W^\Lambda, R^\Lambda, M^\Lambda, \mathbf{V}^\Lambda \rangle$ a Canonic Model (as in Definition 35). Then, for any $w \in W^\Lambda$, $w \Vdash \varphi$ iff $\varphi \in w$.*
Proof: We make an induction on ϕ.

1. *If φ is an atomic formula, it holds by the definition of $\mathbf{V}^\Lambda$.*

2. *If φ is $\neg\phi$ then $w \Vdash \varphi$ iff $\phi \notin w$ by the definition of canonic model.*

3. *If φ is in the form $\phi_1 \wedge \phi_2$ then, $w \Vdash \varphi$ iff $w \Vdash \phi_1$ and $w \Vdash \phi_2$, and $\phi_1 \in w$ and $\phi_2 \in w$.*

4. *If φ is $\langle s, \pi \rangle \phi$, then, $w \Vdash \varphi$ iff:*

   ($\mathbf{R}_\epsilon$): *$f(s,\pi) = \epsilon$, $wR^\Lambda_\pi w$, $w \Vdash \phi$ and $w \in \mathbf{V}^\Lambda(\phi)$ by the inductive hypothesis;*

   (PC): *$\exists u \exists v, wR^\Lambda_{\pi_i} u$, $uR^\Lambda_\pi v$, $1 \leq i \leq n$, $v \Vdash \phi$ and $v \in \mathbf{V}^\Lambda(\phi)$ by the inductive hypothesis and by Lemma 37, where $\pi = \pi_1 \odot \cdots \odot \pi_n$.*

So, this lemma holds. ■

Lemma 39 *If $\varphi \in w$ for all w maximal consistent set of formulae, then $\Vdash \varphi$.*
Proof: Suppose $\nVdash \varphi$; then, by Lemma 38, $\neg\varphi \in w$. But if $\varphi \in w$ and $\neg\varphi \in w$ then we have a contradiction. ■

Theorem 40 *Completeness*

If $\Vdash \varphi$ then $\vdash \varphi$.

Proof: If φ is valid then it is valid in all models, including the canonic. So, it is valid in all worlds of $\mathcal{C}^\Lambda$ (all maximal consistent sets). So by Lemma 39, φ is derivable. Therefore if $\Vdash \varphi$, then $\vdash \varphi$. ■

Definition 41 The Fischer-Ladner closure

It is inductively defined as follows, where $FL(\varphi)$ denotes the smallest set containing φ which is closed under sub formulae.

$FL\colon \Upsilon \to 2^\Upsilon$, where Υ is the set of all formulae

1. *$FL(\varphi)$ is closed under subformulae;*

2. *if $\langle s, \eta \rangle \psi \in FL(\varphi)$, then $\langle s, \eta_i \rangle \langle s_i, \eta \rangle \psi \in FL(\varphi)$,*
   where $\eta = \eta_1 \odot \eta_2 \odot \cdots \odot \eta_n$ and $s_i = f(s, \eta_i)$, for all $1 \leq i \leq n$.

Lemma 42 *Let $\eta = \eta_1 \odot \eta_2 \odot \cdots \odot \eta_n$ be a composed Petri Net program where each η_i, $1 \leq i \leq n$, is a basic Petri Net program and s is a sequence of names. For every sequence $(s_0, \eta) \rightarrow (s_1, \eta) \rightarrow \cdots \rightarrow (s_k, \eta)$, where $s_0 = s$, $k \geq 0$ and $s_j = f(s, \eta_i)$ for some $1 \leq i \leq n$, then either $s_k = \epsilon$ or one of $s_j = s_\ell$ for $0 < j \leq k$, $0 < \ell \leq k$ and $j \neq \ell$.*
Proof: As η has no places that accumulates tokens infinitely, after firing all the basic Petri Net programs of η on and on there are two possibilities for the markup of η:

1. *there is no transition able to fire, so $s_k = \epsilon$;*
2. *some markup m of η activated a loop in the Petri Net (e.g. η as a graph is cyclic), so after a non empty serie of fires the markup m of η will appear again, so $s_j = s_\ell$ for some $0 < j \leq k$ and some $0 < \ell \leq k$ such that $j \neq \ell$.*

■

Lemma 43 *$FL(\varphi)$ is finite.*
Proof: The only possibility of construct $\varphi \succ \sigma$ (i.e. σ is derived from the formula φ) is iff φ is in the form $\langle s, \pi \rangle \Psi$ and σ is in the form $\langle s, \pi_i \rangle \langle s_i, \pi \rangle \Psi$ for some $1 \leq i \leq n$ where n is the size of the Petri Net program π (i.e. the number of atomic programs of π) and π_i is an atomic program. Then the smallest closed set Γ containing a formula ρ is obtained by closing $FL(\rho)$ under $\succ$; hence $\phi \in \Gamma$ iff there is a finite sequence of the form $\varphi = \varphi_1 \succ \cdots \succ \varphi_j = \phi$, where $\forall_{m \neq n} \varphi_m \neq \varphi_n$ and $\varphi \in FL(\rho)$. So, if $\langle s, \kappa \rangle \Psi \succ \langle p, \tau \rangle \phi$ for κ a normalised Petri Net program, then τ is an atomic Petri Net program or is equal to κ. Therefore there can be no infinitely-long $\succ$-sequences.
So, $FL(\varphi)$ is finite. ■

Lemma 44

(i) If $\sigma \in FL(\varphi)$, then $FL(\sigma) \subseteq FL(\varphi)$

(ii) If $\sigma \in FL(\langle s, \pi \rangle \varphi)$, then $FL(\sigma) \subseteq FL(\langle s, \pi \rangle \varphi) \cup FL(\varphi)$

Proof: We make an induction on $FL(\phi)$.

(i) If $\sigma \in FL(\varphi)$, then as by 1. in Definition 41 $FL(\varphi)$ is closed under subformulae. There are three cases depending on the form of φ.

(a) If φ is an atomic proposition p then according to Definition 41 $\sigma = p$ and $FL(\sigma) = FL(\varphi)$.

(b) If φ is in the form $\neg\phi$ then according to Definition 41 $\sigma = \neg\phi$ and by induction and (i) $FL(\sigma) \subseteq FL(\varphi)$.

(c) If φ is in the form $\phi \wedge \psi$ then according to Definition 41 either $\sigma = \phi \wedge \psi$, or $\sigma \in FL(\phi)$ or $\sigma \in FL(\psi)$. In the first case $FL(\sigma) = FL(\varphi)$; in the second and third cases we have $FL(\sigma) \subseteq FL(\phi)$ and $FL(\sigma) \subseteq FL(\psi)$, respectively, by induction and (i). By Definition 41 in either case $FL(\sigma) \subseteq FL(\varphi)$.

Then $FL(\sigma) \subseteq FL(\varphi)$

(ii) If $\sigma \in FL(\langle s, \pi\rangle\varphi)$, then by 2. in Definition 41 $\sigma \in FL(\langle s, \pi\rangle\varphi)$ or $\sigma \in FL(\varphi)$. In the former case, $FL(\sigma) \subseteq FL(\langle s, \pi\rangle\varphi) \cup FL(\varphi)$, by induction and (ii). In the latter case, $FL(\sigma) \subseteq FL(\varphi)$ by induction and (i). Thus in either case, $FL(\sigma) \subseteq FL(\langle s, \pi\rangle\varphi)$ by 2. in Definition 41. Assuming that (i) and (ii) hold for subexpressions, we have three cases (let $\pi = \pi_1 \odot \pi_2 \odot \cdots \odot \pi_n$).

(a) If $\sigma = \langle s, \pi\rangle\varphi$ then $FL(\sigma) = FL(\langle s, \pi\rangle\varphi) \cup FL(\varphi)$ by 2. in Definition 41.

(b) If $\sigma \in FL(\langle s, \pi_i\rangle\langle f(s, \pi_i), \pi\rangle\varphi)$ then $FL(\sigma) \subseteq FL(\langle s, \pi_i\rangle\langle f(s, \pi_i), \pi\rangle\varphi) \cup FL(\langle f(s, \pi_i), \pi\rangle\varphi)$ by induction and (ii) and $FL(\sigma) \subseteq FL(\langle s, \pi_i\rangle\langle f(s, \pi_i), \pi\rangle\varphi) \cup FL(\langle f(s, \pi_i), \pi\rangle\varphi) \cup FL(\varphi)$ by 2. in Definition 41. Hence $FL(\sigma) \subseteq FL(\langle s, \pi\rangle\varphi) \cup FL(\varphi)$ by 2. in Definition 41.

So $FL(\sigma) \subseteq FL(\langle s, \pi\rangle\varphi) \cup FL(\varphi)$.

■

3.4 Computational complexity

In this section we present a polynomial reduction of a well-know EXPTime-hard problem to Petri-PDL SAT: the two person corridor tiling game. This proof is base on another proof presented in the work of Blackburn et al. (2001).

In the two person corridor tiling game, two players (Eloise and Abelard) must place square tiles in a finite grid, where each tile side may have a different color in each side, so that the colours of each side of the tiles match. Each player begins with a finite amount of tiles (with colours randomly defined) and the begin of the grid has a special colour (says white) and there is a special tile for Eloise where, if it is put on column 1 then Eloise wins. When the game begins

Eloise should put a tile in the column 0; in his turn, Abelard must place a tile in the following position on the grid. After the end of the row (for an instance n of the game, the game has n columns), the player must place a tile in the next row, column 0. If no player is able to make a valid move or there are no tiles then Abelard wins. The objective of this game is determine whether Eloise has a winning strategy.

Lemma 45 *The satisfiability of Petri-PDL is EXPTime-hard.*
Proof: The proof goes by reducing the two person corridor tiling problem to the Petri-PDL satisfiability problem, following the methodology of Blackburn et al. (2001).

Given an instance $\mathcal{T} = (n, \{T_0, \ldots, T_{s+1}\})$ of the two person corridor tiling game where n is the width of the corridor and T_i are the tiling types, we will construct a formula φ^τ such that

(i) If Eloise has a winning strategy, φ^τ is satisfiable at the root of some game tree for $\mathcal{T}$ (viewed as a regular Petri-PDL model).

(ii) If φ^τ is satisfiable, then Eloise has a winning strategy in the game $\mathcal{T}$.

(iii) The formula φ^τ can be generated polynomially sized regarding n and s.

The formula φ^τ describes the game tree and states necessary and sufficient conditions for Eloise to win. To construct φ^τ, we will use the following proposition letters:

$\mathbf{t_0}, \ldots, \mathbf{t_{s+1}}$ *to represent the tiles, where t_0 is white;*

$\mathbf{p_1}, \ldots, \mathbf{p_n}$ *to indicate where the tile must be placed in the current round;*

$\mathbf{c_i(t)}, \mathbf{0 \leq i \leq n+1}, \forall \mathbf{t} \in \{\mathbf{t_0}, \ldots, \mathbf{t_{s+1}}\}$ *to indicate the type t of previously placed tile in column i;*

$\mathbf{w}$ *to indicate that the current position is a winning position for Eloise.*

The general schema of a Petri Net (mapped in a Petri Net program η) which models this game is in Figure 3.1, where there is one transition similar to $R^\star$ for each row r such that $1 < r < n$ to denotes that a new row has begun. This Petri Net is constructed as an induction into the definition of the problem. The places denote the states of the game (i.e. after each move the other player may make a move) and each transition is controlling the flow from states.

The sequence s denotes the initial markup of the Petri Net, that is, one token in Row_1 and the tokens needed to denote the initial set of pieces of Eloise and Abelard, according to the legend below. Each place is described bellow.

EC *Eloise can play*

EH *Eloise has piece*

EP *Eloise plays*

AC *Abelard can play*

AH *Abelard has piece*

AP *Abelard plays*

Col$_1$**,. . . ,Col**$_n$ *Each column of the game*

Row$_1$**,. . . ,Row**$_n$ *Each row of the game*

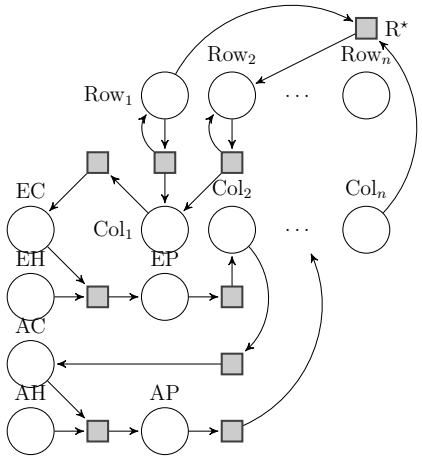


Figure 3.1: Schema of a Petri Net for tiling game

Then, the beginning of the game is described as $[s,\eta]e \wedge p_1 \wedge c_0(\text{white}) \wedge c_1(t_{I_1}) \wedge \cdots \wedge c_n(t_{I_n}) \wedge c_{n+1}(\text{white})$*, where* $I_i, 1 \leq i \leq n$ *denotes the initial tiles and* e *denotes that it is Eloise's turn and* $f(s,\eta) \neq \epsilon$*.*

The set of formulae that rules the game is below.

(i) If Eloise does not have a piece to play at that moment, it is the turn of Abelard: $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)) \rightarrow [s',\eta]\phi$*, where* s' *is a sequence which differs from* s *only by replacing the tokens EC and AC, and EH and AH;*

(ii) If Abelard does not have a piece to play at that moment, it is the turn of Eloise: $(([s',\eta]\phi \leftrightarrow \phi) \wedge \neg([s,\eta]\phi \leftrightarrow \phi)) \rightarrow [s,\eta]\phi$ *(from now on all formulae have omitted a disjunction with a formula* ρ *such that* ρ *differs from the corresponding formula only for changing* s *to* s' *in this enumeration of rules);*

(iii) The referee has already placed white tiles in the columns 0 *and* $n+1$*:* $[s,\eta]col_0(\text{white}) \wedge col_{n+1}(\text{white})$*;*

(iv) The players must respect tiles colors: $C(t',t,t'') \leftrightarrow \text{right}(t') = \text{left}(t)$ *and* $\text{down}(t') = \text{up}(t'')$ *(e.g.* C *holds if the tile* t *can be placed on the right of* t' *and above* t''*, where* t*,* t' *and* t'' *are the positions correspondents to* T*,* T' *and* T''*);*

(v) Ensures tile matching left and downwards: $[s,\eta]((p_i \wedge c_{i-1}(t') \wedge c(t'')) \rightarrow [s',\eta]\bigvee\{c_i(t) \mid C(t',t,t'')\})$*, where* $\bigvee\{c_i(t) \mid C(t',t,t'')\}$ *is the disjunction of all* $C(t',t,t'')$*,* $0 \leq i \leq n$ *and, by convention,* $\bigvee \emptyset = \bot$

(vi) Ensures matching of tiles placed on column n *with white corridor:* $[s,\eta](p_n \rightarrow [s',\eta]\bigvee\{c_n(t) \mid \text{right}(t) = \text{white}\})$

(vii) The first position is a winning position for Eloise: w*.*

As Eloise has a winning strategy, then: $[s,\eta](w \rightarrow (c_1(t_{s+1}) \vee ([s',\eta]\neg w) \vee ([s',\eta]w)))$.

To ensure that the game is finite (e.g. if the game has no end, Abelard wins), the game is limited to $N = n^{s+2}$ *steps with no repetition, so:* $[s,\eta](\text{counter} = N) \rightarrow [s',\eta]\neg w$.

We define φ^τ *as the conjunction of all these formulae.*

If Eloise has a winning strategy, then there is a game tree such that φ^τ *is satisfiable at the root of the game tree viewed as a Petri-PDL model. So, if Eloise has a winning strategy, she can win in at most* N *steps. For a Petri-PDL model* $\mathcal{M}$ *that corresponds to this at-most-*N *steps strategy it is straightforward to check* φ^τ *satisfiability at the root of* $\mathcal{M}$*.*

If $\mathcal{M}, v \Vdash \varphi^\tau$*, then Eloise has a winning strategy, encoded in* $\mathcal{M}$*, in the game* T*. As* w *is satisfied in* v*, Eloise can keep moving through winning positions that she is always able to choose. Hence if* $\text{counter} = N$ *(e.g. the counter has reached),* $[s,\eta]\neg w$ *is satisfied, so there are no more winning positions, but as* $c_1(t_{s+1})$ *is satisfied, so the winning tile was placed in the first step and Eloise has already won.*

So, in a model $\mathcal{M} = \langle W, R_\eta, M, \mathbf{V}\rangle$ *expressing an instance of the two-person corridor tiling game we have two possibilities in a world* u*.*

1. *If Eloise wins the game then* φ^τ *is true in* u*. As* φ^τ *is true it is necessary that all of the following formulae are valid in* u*.*

   (a) $[s,\eta]e \wedge p_1 \wedge c_0(\text{white}) \wedge c_1(t_{I_1}) \wedge \cdots \wedge c_n(t_{I_n}) \wedge c_{n+1}(\text{white})$*, where* $I_i, 1 \leq i \leq n$

   According to the firing function $[s,\eta]e$ *must be true in* u*, as* $f(s,\eta) \neq$

ϵ. *(1)*
$c_0(\text{white}) \wedge c_1(t_{I_1}) \wedge \cdots \wedge c_n(t_{I_n}) \wedge c_{n+1}(\text{white})$ *denotes that the limit tiles are white, as defined in the game where the length of the corridor is n (a generic instance), so it is true in u (2)*
As (1) and (2) are true, then the formula is true.

(b) $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)) \rightarrow [s',\eta]\phi$
For this formula there are two cases.

(i) if $f(s,\eta) = \epsilon$ *then* $f(s',\eta) \neq \epsilon$ *according to* η *definition and the firing function definition (1)*
So $[s',\eta]\phi$ *is true in u (2)*
Then, by (2), $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)) \rightarrow [s',\eta]\phi$ *is true in u.*

(ii) if $f(s,\eta) \neq \epsilon$ *then* $f(s',\eta) = \epsilon$ *according to* η *definition and the firing function definition (1)*
Then $[s',\eta]\phi$ *is true in u. (2)*
As $[s',\eta]\phi$ *is true in u by (2), then* ϕ *is true in all v such that* $uR_\eta v$. *(3)*
According to (1) $[s,\eta]\phi$ *is not true, so there is some world v such that* $uR_\eta v$ *where* ϕ *is false in u. (4)*
By (3) and (4) $([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)$ *is not true in u. (5)*
So, $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)) \rightarrow [s',\eta]\phi$ *is true in u.*

So, $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)) \rightarrow [s',\eta]\phi$ *is true in u.*

(c) $(([s',\eta]\phi \leftrightarrow \phi) \wedge \neg([s,\eta]\phi \leftrightarrow \phi)) \rightarrow [s,\eta]\phi$
For this formula there are two cases.

(i) if $f(s',\eta) = \epsilon$ *then* $f(s,\eta) \neq \epsilon$ *according to* η *definition and the firing function definition (1)*
So $[s,\eta]\phi$ *is true in u (2)*
Then, by (2), $(([s',\eta]\phi \leftrightarrow \phi) \wedge \neg([s,\eta]\phi \leftrightarrow \phi)) \rightarrow [s,\eta]\phi$ *is true in u.*

(ii) if $f(s',\eta) \neq \epsilon$ *then* $f(s,\eta) = \epsilon$ *according to* η *definition and the firing function definition (1)*
Then $[s,\eta]\phi$ *is true in u. (2)*
As $[s,\eta]\phi$ *is true in u by (2), then* ϕ *is true in all v such that* $uR_\eta v$. *(3)*
According to (1) $[s',\eta]\phi$ *is not true, so there is some world v such that* $uR_\eta v$ *where* ϕ *is false in u. (4)*
By (3) and (4) $([s',\eta]\phi \leftrightarrow \phi) \wedge \neg([s,\eta]\phi \leftrightarrow \phi)$ *is not true in u.*

(5)
So, $(([s', \eta]\phi \leftrightarrow \phi) \wedge \neg([s, \eta]\phi \leftrightarrow \phi)) \rightarrow [s, \eta]\phi$ *is true in u.*

(d) $[s, \eta]col_0(\text{white}) \wedge col_{n+1}(\text{white})$
In order to begin the game the referee must place the white tites in the first and last column, so $[s, \eta]col_0(\text{white})$*, which describes the first column white tiles, is true in u. (1)*
In the same way $col_{n+1}(\text{white})$ *which describes the last column white tiles, is true in u. (2). By (1) and (2) we have that* $[s, \eta]col_0(\text{white}) \wedge col_{n+1}(\text{white})$ *is true in u.*

(e) $C(t', t, t'') \leftrightarrow (\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T'')))$ *(this sentence and w are true iff the tiles are well placed)*
For this formula we have two cases.

(i) $C(t', t, t'')$ *is true in u and* $\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T''))$ *is false in u If* $\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T''))$ *is false then we have two possibilities.*

- *If* $\text{right}(t') = \text{left}(T)$ *is false then the side tile do not match and the new tile cannot be placed, so* $C(t', t, t'')$ *is false, then there is a contradiction. (1)*

- *If* $(\text{down}(T') = \text{up}(T''))$ *is false then the bottom tile do not match and the new tile cannot be placed, so* $C(t', t, t'')$ *is false, then there is a contradiction. (2)*

(ii) $C(t', t, t'')$ *is false in u and* $\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T''))$ *is true in u*
If $\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T''))$ *is true in u then* $C(t', t, t'')$ *is true in u, so there is a contradiction. (3)*

So, by (1), (2) and (3), $C(t', t, t'') \leftrightarrow (\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T')))$ *is true in u.*

(f) $[s, \eta]((p_i \wedge c_{i-1}(t') \wedge c(t'')) \rightarrow [s', \eta] \bigvee \{c_i(t) \mid C(t', t, t'')\})$*, where* $0 \leq i \leq n$ *and, by convention,* $\bigvee \emptyset = \bot$ *Suppose this formula is false in u.*
So, $[s, \eta]((p_i \wedge c_{i-1}(t') \wedge c(t''))$ *is true in u (1)*
and $[s', \eta] \bigvee \{c_i(t) \mid C(t', t, t'')\})$ *is false in u. (2)*
By (2) there is some v such that $uR_\eta v$ *where no* $c_i(t)$ *is true. (3)*
So by (3) in v we have that $c_{i-1}(t') \wedge c(t'')$ *is false. (4)*
By (4) $[s, \eta]((p_i \wedge c_{i-1}(t') \wedge c(t''))$ *cannot be true in u, so there is a contradiction. Then, this formula is true.*

(g) $[s,\eta](\text{counter} = N) \rightarrow [s',\eta]\neg w$ *Suppose this formula is false in the world* u.
So $[s,\eta](\text{counter} = N)$ *is true in the world* u *(1), when there are moves available (e.g. some proposition denoting an specific move is true)*
and $[s',\eta]\neg w$ *is false in the world* u. *(2)*
By (2) there is some world v *such that* $uR_\eta v$ *where the formula* w *is false. (3)*
So by (3) in v $\text{counter} = N$ *is false (e.g. the game step counter has not reached its maximum value as long as it is still possible to place tiles). (4)*
By (4) $[s,\eta](\text{counter} = N)$ *is false in the world* u *so there is a contradiction. Then, this formula is true.*

2. *If Eloise loses the game then* φ^τ *is false. if* φ^τ *is false then some* φ^τ *subformulae is false.*

   (a) $[s,\eta]e \wedge p_1 \wedge c_0(\text{white}) \wedge c_1(t_{I_1}) \wedge \cdots \wedge c_n(t_{I_n}) \wedge c_{n+1}(\text{white})$, *where* $I_i, 1 \leq i \leq n$
   To this formula be false in u *we have that*
   Eloise has no tile to place $[s,\eta]e$ *is false in* u, *according to the firing function* $f(s,\eta) = \epsilon$, *(1)*
   or there are no limitant white tiles, so $c_0(\text{white}) \wedge c_1(t_{I_1}) \wedge \cdots \wedge c_n(t_{I_n}) \wedge c_{n+1}(\text{white})$ *is false in* u *(2)*
   By (1) or (2) φ^τ *is false.*

   (b) $(([s,\eta]\phi \leftrightarrow \phi) \wedge \neg([s',\eta]\phi \leftrightarrow \phi)) \rightarrow [s',\eta]\phi$
   If there are no valid tiles to place, then $f(s,\eta) = \epsilon$ *and* $f(s',\eta) = \epsilon$, *according to the firing function. (1)*
   By (1) and axiom R_ϵ, ϕ *is true in* u. *(2) If* ϕ *is true in* u *then some tile must be placed, so* φ^τ *is false.*

   (c) $(([s',\eta]\phi \leftrightarrow \phi) \wedge \neg([s,\eta]\phi \leftrightarrow \phi)) \rightarrow [s,\eta]\phi$
   If there are no valid tiles to place, then $f(s',\eta) = \epsilon$ *and* $f(s,\eta) = \epsilon$, *according to the firing function. (1)*
   By (1) and axiom R_ϵ, ϕ *is true in* u. *(2) If* ϕ *is true in* u *then some tile must be placed, so* φ^τ *is false.*

   (d) $C(t',t,t'') \leftrightarrow (\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T'')))$
   For this formula be false we have that an invalid tile was placed.
   So $C(t',t,t'')$ *is true in* u *(e.g. the tile was placed) (2) and* $\text{right}(t') = \text{left}(T) \wedge (\text{down}(T') = \text{up}(T''))$ *is false in* u *(e.g. the placed tile does not match). So,* φ^τ *will be false.*

(e) $[s,\eta]((p_i \wedge c_{i-1}(t') \wedge c(t'')) \rightarrow [s',\eta]\bigvee\{c_i(t) \mid C(t',t,t'')\})$, *where* $0 \leq i \leq n$ *and, by convention,* $\bigvee\emptyset = \bot$ *For this formula be false we have that an invalid tile was placed.*
So, $[s,\eta]((p_i \wedge c_{i-1}(t') \wedge c(t''))$ *is true in* u *(1)*
and $[s',\eta]\bigvee\{c_i(t) \mid C(t',t,t'')\})$ *is false in* u*. (2)*
As there is an invalid tile then using (2) we have that there is some v *such that* $uR_\eta v$ *where no* $c_i(t)$ *is true. (3)*
So by (3) in v *we have that* $c_{i-1}(t') \wedge c(t'')$ *is false. (4)*
By (4) $[s,\eta]((p_i \wedge c_{i-1}(t') \wedge c(t''))$ *is not true in* u*, so this formula will be false.*

(f) $[s,\eta](\text{counter} = N) \rightarrow [s',\eta]\neg w$ *Suppose this formula is false in the world* u*.*
So $[s,\eta](\text{counter} = N)$ *is true in the world* u *(e.g. the counter has reached the maximum value) (1)*
and $[s',\eta]\neg w$ *is false in the world* u *(e.g. Eloise is not in a winning position). (2)*
By (2) there is there is some world v *such that* $uR_\eta v$ *where the formula* w *is false. (3)*
As by (1) $[s,\eta](\text{counter} = N)$ *is true in the world* u *and by (2)* $[s',\eta]\neg w$ *then this formula will be false in the world* u*.*

As it is possible to encode any $m \geq 2$ *in* $O(\log m+1)$ *binary digits,* N *can be encoded in* $O(\log n^{s+2})$*, which corresponds to* $(s+2)\log n \leq (s+2)n$*. Then, the formula that models the game is polynomial in* s *and* n*. So, the two person corridor tiling problem is polynomially reducible to the Petri-PDL satisfiability problem. Hence, Petri-PDL satisfiability is EXPTime-hard.* ■

3.5
A Natural Deduction system for Petri-PDL

Based on other Natural Deduction systems for Modal Logics (Alechina et al., 2001; Medeiros, 2006; Prawitz, 2006; Simpson, 1994) we define a Natural Deduction system for Petri-PDL.

In all the rules a formula is preceded by a world where it is being considered true (e.g. $w : \varphi$, where w is a world of some model and φ is a well formed formula).

$$\frac{w:\varphi \qquad w:\psi}{w:\varphi\wedge\psi}\wedge\text{-}i \qquad \frac{w:\varphi\wedge\psi}{w:\varphi}\wedge\text{-}e_1 \qquad \frac{w:\varphi\wedge\psi}{w:\psi}\wedge\text{-}e_2 \tag{3-1}$$

$$\dfrac{\dfrac{\begin{array}{c}\{w:\neg\varphi\}^j\\ \vdots\\ w:\bot\end{array}}{w:\varphi}\,\bot_c^j \qquad \dfrac{w:\bot}{w:\varphi}\,\bot_i \tag{3-2}$$

$$\dfrac{\begin{array}{c}\{w:\varphi\}^j\\ \vdots\\ w:\psi\end{array}}{w:\varphi\to\psi}\,{\to}{-}i^j \qquad \dfrac{w:\varphi\to\psi \qquad w:\varphi}{w:\psi}\,{\to}{-}e \tag{3-3}$$

$$\dfrac{w:\varphi}{w:\varphi\vee\psi}\,{\vee}{-}i_1 \qquad \dfrac{w:\psi}{w:\varphi\vee\psi}\,{\vee}{-}i_2 \tag{3-4}$$

$$\dfrac{w:\varphi\vee\psi \qquad \begin{array}{c}\{w:\varphi\}^j\\ \vdots\\ u:\sigma\end{array} \qquad \begin{array}{c}\{w:\psi\}^\ell\\ \vdots\\ u:\sigma\end{array}}{u:\sigma}\,{\vee}{-}e^{j,\ell} \tag{3-5}$$

Note that the "$\to$" (3-3) introduction and "$\bot$" (3-2) and "$\vee$" (3-5) elimination rules may discharge hypothesis (discharges are always identified by "$\{\cdot\}$" and the indexes identify the inference responsible for the discharge).

The inference rules to deal with programs (program rules) have restrictions according to the firing function, where π is any Petri Net program, R_π is the relation indexed to the program π. The discharge of hypothesis is always identified by curly braces (i.e. "$\{\cdot\}''$) indexed by the rule used in it.

$$\dfrac{w:\langle s,\pi\rangle\varphi}{w:\langle s,\pi_1\rangle\langle f(s,\pi_1),\pi\rangle\varphi\vee\cdots\vee\langle s,\pi_n\rangle\langle f(s,\pi_n),\pi\rangle\varphi}\,f_1 \tag{3-6}$$

$$\dfrac{w:[s,\pi]\varphi \qquad wR_\pi u}{u:[f(s,\pi_i),\pi]\varphi}\,f_2 \tag{3-7}$$

$$\dfrac{w:\langle s,\pi\rangle\varphi}{w:\varphi}\,\epsilon_1 \qquad \dfrac{w:\langle s,\pi\rangle\varphi}{wR_\pi w}\,\epsilon_2 \tag{3-8}$$

$$\dfrac{u:\varphi \qquad wR_\pi u}{w:\langle s,\pi\rangle\varphi}\,\pi_\diamond{-}i \tag{3-9}$$

$$\dfrac{w:\langle s,\pi\rangle\varphi \qquad \begin{array}{c}\{u:\varphi\}^j \qquad \{wR_\pi u\}^\ell\\ \vdots\\ v:\psi\end{array}}{v:\psi}\,\pi_\diamond{-}e^{j,\ell} \tag{3-10}$$

$$\dfrac{w:\varphi_1,\ldots,w:\varphi_n \qquad \begin{array}{c}\{w:\varphi_1\}^{j_1},\ldots,\{w:\varphi_n\}^{j_n}\{wR_\pi u\}^\ell\\ \vdots\\ u:\psi\end{array}}{w:[s,\pi]\psi}\,\pi_\Box{-}i^{j_1,\ldots,j_n,\ell} \tag{3-11}$$

$$\frac{w : [s,\pi]\varphi \qquad wR_\pi u}{u : \varphi}\ \pi_{\Box}-e \tag{3-12}$$

The program rules "f_1" (3-6) and "f_2" (3-7), that may already denote formulae with composed Petri Net programs using the definition where $\pi = \pi_1 \odot \pi_2 \odot \cdots \odot \pi_n$ with the restriction that in these rules $f(s,\pi) \neq \epsilon$. Although the "ϵ" (3-8) can be applied only if $f(s,\pi) = \epsilon$.

In the firing (3-6), "$\pi_\diamond$" (3-9) (3-10) and in the the "$\pi_\Box$" (3-11) (3-12) rules it it is important to notice that the sequence s in the program modality introduced must ensure that $f(s,\pi) \neq \epsilon$ unless the world be reflexive and that the world u must be the point where the program stops.

3.5.1
Soundness of the Natural Deduction system for Petri-PDL

We prove the soundness of the Natural Deduction system in the same way of van Dalen (2008). A formula φ can be derived from Γ is true iff it may be achieved by a sequence of inference rules $\mathcal{D}$ of φ with all hypotheses on Γ. So, it suffices to show that the semantical notion of satisfiablity is preserved (i.e. if $\Gamma \vdash \varphi$ then $\Gamma \Vdash \varphi$ in the sense that Γ is a set of assumptions used for the derivation of φ). Let $\Gamma \Vdash_w \varphi$ "φ is true with the hypotheses Γ in a world w."

Lemma 46 *If $\Gamma \vdash \varphi$ then $\Gamma \Vdash \varphi$: the Natural Deduction system for Petri-PDL is sound.*

Proof: The propositional operators are standard into the modal logic literature. We use induction on $\mathcal{D}$.

(base) If the derivation has only one element, then obviously $w : \varphi \in \Gamma$, so it is straightforward that $\Gamma \Vdash_w \varphi$.

(f) 1. *Induction hypothesis: for any Γ containing all hypotheses of $\overset{\mathcal{D}}{w : \langle s,\varphi\rangle\varphi}$, we have that $\Gamma \Vdash_w \langle s,\pi\rangle\varphi$ and $f(s,\pi) \neq \epsilon$. Consider a Γ containing all hypotheses of a $\dfrac{\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi}}{w : \langle s,\pi_1\rangle\langle f(s,\pi_1),\pi\rangle\varphi \vee \cdots \vee \langle s,\pi_n\rangle\langle f(s,\pi_n),\pi\rangle\varphi}$. Let all $\phi \in \Gamma$ be valid, then $\langle s,\pi\rangle\varphi$ is valid in w; so $\langle s,\pi_1\rangle\langle f(s,\pi_1),\pi\rangle\varphi \vee \cdots \vee \langle s,\pi_n\rangle\langle f(s,\pi_n),\pi\rangle\varphi$ is also valid in w, then $\Gamma \Vdash_w \langle s,\pi_1\rangle\langle f(s,\pi_1),\pi\rangle\varphi \vee \cdots \vee \langle s,\pi_n\rangle\langle f(s,\pi_n),\pi\rangle\varphi$.*

2. *Induction hypothesis: $\overset{\mathcal{D}}{w : [s,\pi]\varphi}$ and $\overset{\mathcal{D}'}{wR_\pi u}$ are derivations and for each Γ, Γ' containing the hypotheses of $\mathcal{D}, \mathcal{D}'$, $\Gamma \Vdash_w [s,\pi]\varphi$ and*

$\{wR_\pi u\} \subseteq \Gamma'$ *and* $f(s,\pi_i) \neq \epsilon$. *Now let* Γ'' *contain the hypotheses of* $\dfrac{\overset{\mathcal{D}}{w : [s,\pi]\varphi} \qquad \overset{\mathcal{D}'}{wR_\pi u}}{w : [f(s,\pi_i),\pi]\varphi}$. *Choosing* Γ'' *to be the set of hypotheses of* $\mathcal{D}$ *and* $\mathcal{D}'$ *then* $\Gamma'' \supseteq \Gamma \cup \Gamma'$. *So* $\Gamma'' \Vdash_w [s,\pi]\varphi$ *and* $\{wR_\pi u\} \subseteq \Gamma''$. *Let all* $\phi \in \Gamma''$ *be valid, then* $[s,\pi]\varphi$ *is true in* w *and* $wR_\pi u$; *hence* $[f(s,\pi_i),\pi]\varphi$ *is true in* w, *which shows that* $\Gamma'' \Vdash_w [f(s,\pi_i),\pi]\varphi$.

(ϵ*)* *Induction hypothesis: for any* Γ *containing the hypotheses of* $\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi}$ *we have that* $\Gamma \Vdash_w \langle s,\pi\rangle\varphi$ *and* $f(s,\pi) = \epsilon$. *Consider a* Γ *containing all hypotheses of* $\dfrac{\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi}}{w : \varphi}$ *and* $\dfrac{\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi}}{wR_\pi u}$. *Let all* $\phi \in \Gamma$ *be valid, then* $\langle s,\pi\rangle\varphi$ *is true in* w; *so* φ *is true in* w *and* $wR_\pi u$, *then* $\Gamma \Vdash_w \varphi$ *and* $wR_\pi u$.

($\pi_\diamond - i$*)* *Induction hypothesis:* $\overset{\mathcal{D}}{u : \varphi}$ *and* $\overset{\mathcal{D}'}{wR_\pi u}$ *are derivations and for each* Γ, Γ' *containing the hypotheses of* $\mathcal{D}, \mathcal{D}'$, $\Gamma \Vdash_u \varphi$ *and* $\{wR_\pi u\} \subseteq \Gamma'$ *and* $f(s,\pi) \neq \epsilon$. *Now let* Γ'' *contain the hypotheses of* $\dfrac{\overset{\mathcal{D}}{u : \varphi} \qquad \overset{\mathcal{D}'}{wR_\pi u}}{w : \langle s,\pi\rangle\varphi}$. *Choosing* Γ'' *to be the set of hypotheses of* $\mathcal{D}$ *and* $\mathcal{D}'$ *then* $\Gamma'' \supseteq \Gamma \cup \Gamma'$. *So* $\Gamma'' \Vdash_u \varphi$ *and* $\{wR_\pi u\} \subseteq \Gamma''$. *Let all* $\phi \in \Gamma''$ *be valid, then* φ *is true in* u *and* $wR_\pi u$; *hence* $\langle s,\pi\rangle\varphi$ *is true in* w, *which shows that* $\Gamma'' \Vdash_w \langle s,\pi\rangle\varphi$.

($\pi_\diamond - e$*)* *Induction hypothesis: for any* Γ *containing all hypotheses of* $\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi}$ *and* $\overset{u : \varphi \qquad wR_\pi u}{\overset{\mathcal{D}'}{v : \psi}}$ *and* $f(s,\pi) \neq \epsilon$, *then* $\Gamma \Vdash_v \psi$. *Let* Γ' *contain all hypotheses of* $\dfrac{\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi} \qquad \overset{\{w:\varphi\}^j \qquad \{wR_\pi u\}^\ell}{\overset{\mathcal{D}'}{v : \psi}}}{v : \psi}\, j,\ell$. $\Gamma' \cup \{u : \varphi\} \cup \{wR_\pi u\}$ *contains all the hypotheses of* $\dfrac{\overset{u : \varphi \qquad wR_\pi u}{\overset{\mathcal{D}}{w : \langle s,\pi\rangle\varphi}} \qquad \overset{\mathcal{D}'}{v : \psi}}{v : \psi}$ *so* $\langle s,\pi\rangle\varphi$ *is true in* w, φ *is true in* u *and* $wR_\pi u$ *as also all* $\sigma \in \Gamma'$ *are valid, then* ψ *is valid in* v; *hence* $\Gamma' \Vdash_v \psi$.

($\pi_\Box - i$*)* *Induction hypothesis: for any* Γ *containing all hypotheses of* $\overset{\mathcal{D}}{w : \varphi_1, \ldots, w : \varphi_n}$ *and* $\overset{w : \varphi_1, \ldots, w : \varphi_n, wR_\pi u}{\mathcal{D}'}$ $\overset{}{u : \psi}$ *and* u *is the final state of a program* π *(i.e.* π *stops in* u*), then* $\Gamma \Vdash_u \psi$. *Let* Γ' *contain all hypotheses of* $\dfrac{\overset{\mathcal{D}}{w : \varphi_1, \ldots, w : \varphi_n} \qquad \overset{\{w:\varphi_1\}^{j_1}, \ldots, \{\varphi_n\}^{j_n}, \{wR_\pi u\}^\ell}{\overset{\mathcal{D}'}{u : \psi}}}{w : [s,\pi]\psi}\, j_1,\ldots,j_n,\ell$. $\Gamma' \cup \{u : \varphi_1\} \cup \cdots \cup \{\varphi_n\} \cup \{wR_\pi u\}$ *contains all the hypotheses of*

$$\frac{\begin{array}{cc}\mathcal{D} & \mathcal{D}' \\ w:\varphi_1,\ldots,w:\varphi_n & u:\psi\end{array}}{w:[s,\pi]\psi}\quad \begin{array}{c} w:\varphi_1,\ldots,w:\varphi_n, wR_\pi u \\ \text{, so } \varphi_1,\ldots,\varphi_n \text{ are valid} \end{array}$$

in w, ψ is valid in u and $wR_\pi u$ as also all $\sigma \in \Gamma'$ are valid, then $[s,\pi]\psi$ is true in w; hence $\Gamma' \Vdash_w [s,\pi]\psi$.

($\pi_\Box - e$) Induction hypothesis: $\begin{array}{c}\mathcal{D} \\ w:[s,\pi]\varphi\end{array}$ *and* $\begin{array}{c}\mathcal{D}' \\ wR_\pi u\end{array}$ *are derivations and for each Γ, Γ' containing the hypotheses of $\mathcal{D}, \mathcal{D}'$, $\Gamma \Vdash_w [s,\pi]\varphi$ and $\{wR_\pi u\} \subseteq \Gamma'$ and u is the final state of the program π (i.e. π stops in u). Now let Γ'' contain the hypotheses of* $\dfrac{\begin{array}{cc}\mathcal{D} & \mathcal{D}' \\ w:[s,\pi]\varphi & wR_\pi u\end{array}}{u:\varphi}$ *. Choosing Γ'' to be the set of hypotheses of $\mathcal{D}$ and $\mathcal{D}'$ then $\Gamma'' \supseteq \Gamma \cup \Gamma'$. So $\Gamma'' \Vdash_w [s,\pi]\varphi$ and $\{wR_\pi u\} \subseteq \Gamma''$. Let all $\phi \in \Gamma''$ be valid, then $[s,\pi]\varphi$ is true in w and $wR_\pi u$; hence φ is true in u, which shows that $\Gamma'' \Vdash_u \varphi$.*

As all rules preserve the truth, the system is sound. ■

3.5.2
Completeness of the Natural Deduction system for Petri-PDL

To prove the completeness of the system we derive the axioms and show the representation of the operators.

Lemma 47 *The axioms of Petri-PDL are derivable by its Natural Deduction system.*
Proof:

(PL) The non-modal propositional part is complete regarding derivations of $w:\alpha$ from $w:\Gamma$. Thus, all (PL) labeled axioms are provable in Petri-PDL Natural Deduction system.

(K) $[s,\pi](p \to q) \to ([s,\pi]p \to [s,\pi]q)$

$$\dfrac{\dfrac{\dfrac{\{w:[s,\pi](p\to q)\}^3,\ \{w:[s,\pi]p\}^2 \qquad \dfrac{\pi_{\Box-e}\dfrac{\{w:[s,\pi](p\to q)\}^1 \quad \{wR_\pi u\}^1}{u:p\to q} \qquad \dfrac{\{w:[s,\pi]p\}^1 \quad \{wR_\pi u\}^1}{u:p}\,\pi_{\Box-e}}{u:q}\to -e}{w:[s,\pi]q}\,\pi^1_{\Box-i}}{\dfrac{w:[s,\pi]p\to[s,\pi]q}{}}\to -i^2}{w:[s,\pi](p\to q)\to([s,\pi]p\to[s,\pi]q)}\to -i^3$$

(Du) $[s,\pi]\varphi \leftrightarrow \neg\langle s,\pi\rangle\neg\varphi$

For this formula we have two derivations.

(a) $[s,\pi]\varphi \rightarrow \neg\langle s,\pi\rangle\neg\varphi$

$$\dfrac{\dfrac{\pi_{\Box-e}\dfrac{\{w:[s,\pi]\varphi\}^3 \quad \{wR_\pi u\}^3}{u:\varphi} \quad \pi^1_{\Diamond-e}\dfrac{\{w:\langle s,\pi\rangle\neg\varphi\}^2 \quad \{u:\neg\varphi\}^1\{wR_\pi u\}^1}{u:\neg\varphi}}{\bot}}{\dfrac{\dfrac{u:\bot}{w:\neg\langle s,\pi\rangle\neg\varphi}\,\bot^2_c}{w:[s,\pi]\varphi\rightarrow\langle s,\pi\rangle\neg\varphi}\rightarrow -i^3}$$

(b) $\neg\langle s,\pi\rangle\neg\varphi \rightarrow [s,\pi]\varphi$

$$\dfrac{\dfrac{\{w:\neg\langle s,\pi\rangle\neg\varphi\}^3 \quad \dfrac{\dfrac{\pi_{\Diamond-e}\dfrac{\{u:\neg\varphi\}^1 \quad \{wR_\pi u\}^2}{w:\langle s,\pi\rangle\neg\varphi}}{\bot} \quad \{w:\neg\langle s,\pi\rangle\neg\varphi\}^2}{\dfrac{w:\bot}{u:\varphi}\,\bot^1_c}\,\pi^2_{\Box-i}}{w:[s,\pi]\varphi}}{w:\neg\langle s,\pi\rangle\neg\varphi\rightarrow[s,\pi]\varphi}\rightarrow -i$$

(PC) $\langle s,\eta\rangle\varphi \leftrightarrow \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$,
where $s_i = f(s,\eta_i)$, *for all* $1 \leq i \leq n$
For this formula we have two derivations.

(a) $\langle s,\eta\rangle\varphi \rightarrow \langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$
This case is straightforward from the definition of the firing rule (3-6).

(b) $\langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi \rightarrow \langle s,\eta\rangle\varphi$
Due to the lack of space we abbreviate "$\langle s,\eta_1\rangle\langle s_1,\eta\rangle\varphi \vee \langle s,\eta_2\rangle\langle s_2,\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$*" to* Ψ, $\langle s,\eta_j\rangle\langle s_j,\eta\rangle\varphi \vee \langle s,\eta_{j+1}\rangle\langle s_{j+1},\eta\rangle\varphi \vee \cdots \vee \langle s,\eta_n\rangle\langle s_n,\eta\rangle\varphi$ *to* Ψ_{j-1} *and "*$\langle s,\eta_i\rangle\langle s_i,\eta\rangle\varphi$*" to* Ψ'_i.

$$\dfrac{\dfrac{\{w:\Psi\}^{j} \quad \dfrac{\dfrac{\{w:\Psi_1'\}^{2} \quad \pi_{\diamond-i}\dfrac{\{u:\varphi\}^{1} \quad \{wR_\pi u\}^{1}}{w:\langle s,\eta\rangle\varphi}}{w:\langle s,\eta\rangle\varphi}\,\pi^{1}_{\diamond-e} \quad \dfrac{\{w:\Psi_1\}^{2} \quad \dfrac{\dfrac{\{w:\Psi_2'\}^{3} \quad \pi_{\diamond-i}\dfrac{\{u:\varphi\}^{4} \quad \{wR_\pi u\}^{4}}{w:\langle s,\eta\rangle\varphi}}{w:\langle s,\eta\rangle\varphi}\,\pi^{4}_{\diamond-e} \quad \{w:\Psi_2\}^{2}}{w:\langle s,\eta\rangle\varphi}\vee-e^{2} \quad \vdots}{w:\langle s,\eta\rangle\varphi}\vee-e^{3}}{w:\Psi\to\langle s,\eta\rangle\varphi}\to-i^{j}$$

(R_ϵ) $\langle s,\eta\rangle\varphi \leftrightarrow \varphi$, *if* $f(s,\eta) = \epsilon$
For this formula we have two derivations.

(a) $\langle s,\eta\rangle\varphi \rightarrow \varphi$
This case is straightforward from the definition of the ϵ_1 (3-8) *rule.*

(b) $\varphi \rightarrow \varphi\langle s,\eta\rangle$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{w : \varphi} \qquad \dfrac{\{w : \langle s,\eta\rangle\varphi\}^1}{wR_\pi w}\, f_{\epsilon_2}}{w : \langle s,\eta\rangle\varphi}\, \pi_{\diamond -i}}{w : \langle s,\eta\rangle\varphi \rightarrow \langle s,\eta\rangle\varphi}\, \rightarrow -i^1}{w : \langle s,\eta\rangle\varphi}\, \rightarrow -e$$

(Sub) It is straightforward from the rules definition.

(MP) It is straightforward from the rules (3-3).

(Gen) It is straightforward from the rules (3-11).

So, the Natural Deduction system for Petri-PDL can derive its axioms. ■

Theorem 48 *The Natural Deduction system for Petri-PDL is complete.*
Proof: By Lemma 47 the Natural Deduction system can simulate Petri-PDL axiomatic system and the operators behaviour derivation are straightforward from the rules definition, so the system is complete. ■

3.6
A Resolution system for Petri-PDL

In this section we present a clausal Resolution based calculus for Petri-PDL proposed by Nalon et al. (2014). In order to prove that a formula φ is valid, we apply the inference rules to the clausal form of the negated formula, $\neg\varphi$. The transformation into the normal form follows the works of Nalon & Dixon (2006) and Degtyarev et al. (2006), which use anti-prenexing together with simplification, followed by rewriting and renaming to separate the contexts to which the inference rules are applied.

3.6.1 Normal Form

Let φ be a formula in the language of Petri-PDL. The set of inference rules are applied to the transformation of φ into a specific normal form, called *Divided Separated Normal Form for Petri-PDL* (DSNF_{PPDL}), which separates the contexts (formulae which are true only at the initial state; formulae which are true in all states) for reasoning. Before applying the transformation, we require that a formula φ to be in Anti-Prenex Normal Form (APNF), i.e. when modal operators are moved inwards a formula and only applied to modal literals. It has been shown by Egly (1994) that the transformation of a given problem into anti-prenex normal form (i.e. when quantifiers are moved inwards a formula) results in a better set of clauses for First-Order Logics. The same approach for modal logics was investigated by Nalon & Dixon (2006), where it has been shown that anti-prenexing together with simplification may also result in a better set of clauses for a particular logic if its language allows for collapsing and/or simplification of modal operators. The application of such a technique to formulae in the language of Petri-PDL is justified by the axiom (PC), which allows similar simplifications. The transformation rules into APNF are given after the following definitions.

Definition 49 Literal

A literal *is either* $\top$, p *or* $\neg p$, *for* $p \in \Phi$. *For a literal* l *of the form* $\neg p$, *where* p *is a propositional symbol,* $\neg l$ *denotes* p; *for a literal* l *of the form* p, $\neg l$ *denotes* $\neg p$. *The literals* l *and* $\neg l$ *are called* complementary literals. *A* modal literal *is either* $\langle s, \pi\rangle l$ *or* $[s, \pi]l$, *where* s *is a sequence of names,* π *is a Petri Net program, and* l *is a literal.*

Definition 50 Modal term

A modal term *is a formula of the form* $M_0 \cdots M_k l$, *where* l *is a literal and* M_i *is of the form* $[s_i, \pi_i]$ *or* $\langle s_i, \pi_i\rangle$, $0 \leq i \leq k$, $k \in \mathbb{N}$, *where each* s_i *is a sequence of names and* π_i *is a Petri Net program.*

Note that when $k = 0$, the literal l is not preceded by any modal operator.

Definition 51 Anti-Prenex Normal Form (APNF)

Let φ *and* ψ *be formula in the language of Petri-PDL. A formula* χ *is in Anti-Prenex Normal Form (APNF) if, and only if,*

1. χ *is a modal term; or*
2. χ *is of the form* $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, *or* $(\varphi \rightarrow \psi)$, *and* φ *and* ψ *are in APNF;*

3. *χ is of the form $[s,\pi]\varphi$, φ is disjunctive (in the form $\phi_1 \vee \cdots \vee \phi_n$), and φ is in APNF; or*

4. *χ is of the form $\langle s,\pi\rangle\varphi$, φ is conjunctive, and φ is in APNF.*

We define a function $\alpha(\varphi)$, where φ is a formula, which produces the anti-prenex normal form of φ. The base case occurs when the formula φ is already in APNF, that is, φ is a modal term. In this case, $\alpha(\varphi) = \varphi$. If the main operator is modal, it can be distributed over subformulae in the following cases (where φ and ψ are formulae):

$$\begin{aligned}
\alpha([s,\pi](\varphi \to \psi)) &= \alpha([s,\pi]\varphi \to [s,\pi]\psi) \\
\alpha([s,\pi](\varphi \wedge \psi)) &= \alpha([s,\pi]\varphi \wedge [s,\pi]\psi) \\
\alpha([s,\pi]\neg(\varphi \to \psi)) &= \alpha([s,\pi]\varphi \wedge [s,\pi]\neg\psi) \\
\alpha([s,\pi]\neg(\varphi \vee \psi)) &= \alpha([s,\pi]\neg\varphi \wedge [s,\pi]\neg\psi) \\
\alpha(\langle s,\pi\rangle(\varphi \to \psi)) &= \alpha(\langle s,\pi\rangle\neg\varphi \vee \langle s,\pi\rangle\psi) \\
\alpha(\langle s,\pi\rangle(\varphi \vee \psi)) &= \alpha(\langle s,\pi\rangle\varphi \vee \langle s,\pi\rangle\psi) \\
\alpha(\langle s,\pi\rangle\neg(\varphi \wedge \psi)) &= \alpha(\langle s,\pi\rangle\neg\varphi \vee \langle s,\pi\rangle\neg\psi)
\end{aligned}$$

If we have two consecutive modal operators, the function is applied recursively, where φ is of the form $[s',\pi']\psi$ or $\langle s',\pi'\rangle\psi$, for any s' a sequence of names and π' a Petri Net program, and ψ is a formulae which is not in APNF:

$$\alpha([s,\pi]\varphi) = \alpha([s,\pi]\alpha(\varphi)) \qquad \alpha(\langle s,\pi\rangle\varphi) = \alpha(\langle s,\pi\rangle\alpha(\varphi))$$

If the main operator is a modal operator, but the formula inside its scope is not one of the above, we apply the anti-prenexing function to this formula, that is:

$$\alpha([s,\pi]\varphi) = [s,\pi]\alpha(\varphi) \qquad \alpha(\langle s,\pi\rangle\varphi) = \langle s,\pi\rangle\alpha(\varphi)$$

When the main operator is classical, the transformation function is also applied recursively. Note that when the polarity of a subformula is negative, we rewrite the formula in order to make this explicit.

$$\begin{aligned}
\alpha(\neg[s,\pi]\varphi) &= \alpha(\langle s,\pi\rangle\neg\varphi) \\
\alpha(\varphi \to \psi) &= \alpha(\neg\varphi) \vee \alpha(\psi) \\
\alpha(\varphi \wedge \psi) &= \alpha(\varphi) \wedge \alpha(\psi) \\
\alpha(\varphi \vee \psi) &= \alpha(\varphi) \vee \alpha(\psi) \\
\alpha(\neg\langle s,\pi\rangle\varphi) &= \alpha([s,\pi]\neg\varphi) \\
\alpha(\neg(\varphi \to \psi)) &= (\alpha(\varphi) \wedge \alpha(\neg\psi)) \\
\alpha(\neg(\varphi \wedge \psi)) &= (\alpha(\neg\varphi) \vee \alpha(\neg\psi)) \\
\alpha(\neg(\varphi \vee \psi)) &= (\alpha(\neg\varphi) \wedge \alpha(\neg\psi))
\end{aligned}$$

Because of the axiom (PC) and seriality, simplification of modal operators can be applied at any step of the transformation into the anti-prenexing normal form, as follows (where $\pi = \pi_1 \odot \pi_2 \odot \cdots \odot \pi_n$, $1 \leq i \leq n$):

$$\begin{aligned}
\alpha([s,\pi_i][f(s,\pi_i),\pi]\varphi) &= \alpha(\langle s,\pi\rangle\varphi) \\
\alpha([s,\pi_i]\langle f(s,\pi_i),\pi\rangle\varphi) &= \alpha(\langle s,\pi\rangle\varphi) \\
\alpha(\langle s,\pi_i\rangle[f(s,\pi_i),\pi]\varphi) &= \alpha(\langle s,\pi\rangle\varphi) \\
\alpha(\langle s,\pi_i\rangle\langle f(s,\pi_i),\pi\rangle\varphi) &= \alpha(\langle s,\pi\rangle\varphi)
\end{aligned}$$

Note that, at the end of the transformation, $\alpha(\varphi)$ is in both APNF and in Negated Normal Form (that is, a formula where the connectives are restricted to $\neg$, $\vee$, $\wedge$, and the modal operators and the negations are applied only to propositional symbols). The proof that the transformation into APNF is correct and satisfiability preserving is given by the following lemma.

Lemma 52 *Let φ be a formula in the language of Petri-PDL and let $\alpha(\varphi)$ be a formula resulting from the transformation of φ into APNF. If $\Vdash \varphi$, then $\Vdash \alpha(\varphi)$.*

Proof: Suppose φ is a Petri-PDL formula.

1. *If φ is atomic it is straightforward from Petri-PDL language definition.*
2. *If φ is in the form $\neg\phi$ it is straightforward from Petri-PDL language definition.*
3. *If φ is in the form $[s,\pi](\phi \rightarrow \psi)$ it is straightforward from axiom (K).*
4. *If φ is in the form $[s,\pi](\phi \wedge \psi)$ it is straightforward from axiom (K).*
5. *If φ is in the form $[s,\pi]\neg(\phi \rightarrow \psi)$ it is straightforward from the standard implication equivalence by means of conjunction and axiom (K).*
6. *If φ is in the form $[s,\pi]\neg(\phi \vee \psi)$ it is straightforward from the De Morgan rule and axiom (K).*
7. *If φ is in the form $\langle s,\pi\rangle(\phi \rightarrow \psi)$ it is straightforward from axiom (K).*
8. *If φ is in the form $\langle s,\pi\rangle(\phi \vee \psi)$ it is straightforward from axiom (K).*
9. *If φ is in the form $\langle s,\pi\rangle\neg(\phi \wedge \psi)$ it is straightforward from De Morgan rule and axiom (K).*
10. *If φ is in the form $[s,\pi]\phi$ it is straightforward from Petri-PDL language definition.*

11. *If φ is in the form $\langle s, \pi \rangle \phi$ it is straightforward from Petri-PDL language definition.*

12. *If φ is in the form $\neg[s, \pi]\phi$ it is straightforward from axiom (Du).*

13. *If φ is in the form $\neg\langle s, \pi \rangle \phi$ it is straightforward from axiom (Du).*

14. *If φ is in the form $\phi \rightarrow \psi$ it is straightforward from Petri-PDL language definition.*

15. *If φ is in the form $\phi \vee \psi$ it is straightforward from Petri-PDL language definition.*

16. *If φ is in the form $\phi \wedge \psi$ it is straightforward from Petri-PDL language definition.*

17. *If φ is in the form $\neg(\phi \rightarrow \psi)$ it is straightforward from the standard implication equivalence by means of conjunction.*

18. *If φ is in the form $\neg(\phi \wedge \psi)$ it is straightforward from De Morgan rule.*

19. *If φ is in the form $\neg(\phi \vee \psi)$ it is straightforward from De Morgan rule.*

20. *If φ is in the form $[s, \pi_i][f(s, \pi_i), \pi]\phi$ it is straightforward from axiom (PC).*

21. *If φ is in the form $[s, \pi_i]\langle f(s, \pi_i), \pi \rangle \phi$ it is straightforward from axiom (PC).*

22. *If φ is in the form $\langle s, \pi_i \rangle \langle f(s, \pi_i), \pi \rangle \phi$ it is straightforward from axiom (PC).*

23. *If φ is in the form $\langle s, \pi_i \rangle [f(s, \pi_i), \pi]\phi$ it is straightforward from axiom (PC).*

As all the rules preserve the satisfiability, then if $\Vdash \varphi$, then $\Vdash \alpha(\varphi)$ ■

In order to separate the contexts for reasoning, we define a *Petri-PDL problem* to be a tuple $\langle \mathcal{I}, \mathcal{U} \rangle$, where $\mathcal{I}$, the set of initial formulae, is a finite set of non-modal propositional formulae; and $\mathcal{U}$, the set of universal formulae, is a finite set of formulae in the language of Petri-PDL. We will extend the frame model for Petri-PDL by including an initial world $w_0 \in W$. So, let $\mathcal{M} = \langle W, w_0, R_\pi, M, \mathbf{V} \rangle$ be a Petri-PDL model where $w_0 \in W$ is the initial world (an initial state for the Petri Net, corresponding to its initial markup). We say that a Petri-PDL problem $\langle P \rangle = \langle \mathcal{I}, \mathcal{U} \rangle$ is satisfied in $\mathcal{M}$ (denoted by

$\mathcal{M} \Vdash \langle P \rangle$) if and only if $\mathcal{M}$,$w_0 \Vdash \mathcal{I}$ and, for all $w_0 \in W$, $\mathcal{M}$,$w_0 \Vdash \mathcal{U}$ (where satisfiability of sets is defined in the usual way).

Let $\alpha(\varphi)$ be a formula in APNF. The set of resolution-based inference rules, given in Section 3.6.2 are applied to the transformation of $\alpha(\varphi)$ into a *clausal Petri-PDL problem*, which is formally defined as Petri-PDL problem $\langle \mathcal{I},\mathcal{U} \rangle$, where $\mathcal{I}$, the set of initial clauses, contains formulae in the form of $\bigvee_i l_i$, $i \in \mathbb{N}$, where l_i are literals; and $\mathcal{U}$, the set of universal clauses, contains formulae in the form of $\bigvee_i l_i \vee \bigvee_j [s_j, \pi_j] l'_j \vee \bigvee_k \neg [s'_k, \pi'_k] l''_k$, $i, j, k \in \mathbb{N}$, where l_i, l'_j, l''_k are literals, s_i, s_j, s'_k are sequences of names, and π_i, π_j, π'_k are Petri Net programs.

The transformation of a formula φ into the clausal form starts by taking the problem $\langle \{t_0\}, \{t_0 \rightarrow \alpha(\varphi)\} \rangle$, where t_0 is a new propositional symbol (i.e. a propositional symbol that does not occur in φ), and applying exhaustively the following rewriting rules (where t is a literal, t_1 is a new propositional symbol, and ψ_1, ψ_2 are formulae):

(τ_1) $\langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow (\psi_1 \wedge \psi_2)\} \rangle \longrightarrow \langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow \psi_1, t \rightarrow \psi_2\} \rangle$;

(τ_2) $\langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow (\psi_1 \vee \psi_2)\} \rangle \longrightarrow \langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow (\psi_1 \vee t_1), t_1 \rightarrow \psi_2\} \rangle$, if ψ_2 is not a literal;

(τ_3) $\langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow \langle s, \pi \rangle \psi_1\} \rangle \longrightarrow \langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow \langle s, \pi \rangle t_1, t_1 \rightarrow \psi_1\} \rangle$, if ψ_1 is not a literal;

(τ_4) $\langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow [s, \pi] \psi_1 \rangle \longrightarrow \langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow [s, \pi] t_1, t_1 \rightarrow \psi_1\} \rangle$, if ψ_1 is not a literal .

Note that we take conjunctions and disjunctions as being associative and commutative. Thus, for instance, the transformation rule ($\tau_{3.6.1}$) also applies when ψ_1 is not a literal. As a final step, we replace the modal operator $\langle s, \pi \rangle$ by its dual and rewrite implications as disjunctions, that is, we apply the following rewriting rules (where t, l are literals, D is a disjunction of literals and/or modal literals, s is a sequence of names, and π is a Petri Net program):

(τ_5) $\langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow \langle s, \pi \rangle l\} \rangle \longrightarrow \langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow \neg [s, \pi] \neg l\} \rangle$;

(τ_6) $\langle \mathcal{I},\mathcal{U} \cup \{t \rightarrow D\} \rangle \longrightarrow \langle \mathcal{I},\mathcal{U} \cup \{\neg t \vee D\} \rangle$.

Note that simplification takes place at any step of transformation, that is, we remove occurrences of the constants $\top$ and $\bot$ as well as duplicates of formulae in conjunctions and disjunctions. This is achieved by exhaustively applying the following simplification rules (where conjunctions and disjunctions are commutative, φ is a formula, s is a sequence of names, and π is a Petri Net program):

$$
\begin{array}{rcl}
\varphi \wedge \top & \longrightarrow & \varphi \\
\varphi \vee \top & \longrightarrow & \top \\
\varphi \wedge \bot & \longrightarrow & \bot \\
\varphi \vee \bot & \longrightarrow & \varphi \\
\neg\top & \longrightarrow & \bot \\
\neg\bot & \longrightarrow & \top \\
\varphi \vee \varphi & \longrightarrow & \varphi \\
\varphi \wedge \varphi & \longrightarrow & \varphi \\
\varphi \vee \neg\varphi & \longrightarrow & \top \\
\varphi \wedge \neg\varphi & \longrightarrow & \bot \\
[s,\pi]\top & \longrightarrow & \top \\
[s,\pi]\bot & \longrightarrow & \bot \\
\neg\neg\varphi & \longrightarrow & \varphi
\end{array}
$$

Lemma 53 *Let φ be a well-formed formula in the language of Petri-PDL. φ is satisfiable if and only if the transformation of $\alpha(\varphi)$ into DSNF_{PPDL} is satisfiable.*

Proof: This proof is straightforward from Lemma 52 and from the axiomatisation of Petri-PDL. ■

3.6.2 Resolution rules

Let φ be a formula in the language of Petri-PDL and let $\tau(\varphi)$ be the set of clauses resulting from the transformation of φ into its normal form, as given in the previous section. The resolution method for Petri-PDL, named RES_{PPDL}, consists of applying the following inference rules to clauses in $\tau(\varphi)$ (where C, C' are disjunctions of literals, D, D' are disjunctions of literals or modal literals, l, l_i, $0 \leq i \leq n$, are literals, s is a sequence of names, and π, η are Petri Net programs).

$$
\textsf{ires}\ \frac{C \vee l \in \mathcal{I} \cup \mathcal{U} \qquad C' \vee \neg l \in \mathcal{I}}{C \vee C' \in \mathcal{I}} \qquad \textsf{ures}\ \frac{C \vee l \in \mathcal{U} \qquad C' \vee \neg l \in \mathcal{U}}{C \vee C' \in \mathcal{U}} \tag{3-13}
$$

$$
\textsf{ser1}\ \frac{D \vee [s,\pi]l \in \mathcal{U}}{D \vee \neg[s,\pi]\neg l \in \mathcal{U}} \tag{3-14}
$$

$$
\textsf{ser2}\ \frac{D \vee \neg[s,\pi]l \in \mathcal{U} \qquad l_1 \vee \ldots \vee l_n \vee l \in \mathcal{U}}{D \vee \neg[s,\pi]\neg l_1 \vee \ldots \vee \neg[s,\pi]\neg l_n \in \mathcal{U}} \tag{3-15}
$$

$$\begin{array}{ll} \mathsf{ref} & \dfrac{D \ \vee \ [s,\pi]l \ \in \mathcal{U}}{D \ \vee \ l \ \in \mathcal{U}} \\ \text{if } f(s,\pi) = \epsilon & \end{array} \tag{3-16}$$

$$\begin{array}{ll} \mathsf{comp} & \dfrac{D \ \vee \ \neg[s,\pi]l \ \in \mathcal{U} \quad D' \ \vee \ [r,\eta]l \ \in \mathcal{U}}{D \ \vee \ D' \ \in \mathcal{U}} \\ \text{if } \pi \subseteq \eta \text{ and } & \\ f(s,\pi_b) \leftrightarrow f(r,\pi_b) & \end{array} \tag{3-17}$$

The inference rules ires and ures (3-13) are equivalent to classical resolution applied within each context of a given problem. The inference rules ser1 (3-14) and ser2 (3-15) deal with seriality: a Petri Net cannot lead to a contradicting state. The inference rule ref (3-16) corresponds to reflexivity and it can only be applied if $f(s,\pi) = \epsilon$. The last inference rule, comp (3-17), deals with compositionality: if π is a Petri subnet of η (as a graph), then we cannot have that both π and η lead to a contradicting state, through sequences of names s and r where for all basic program of $\pi = \pi_1 \odot \cdots \odot \pi_n$,$f(s,\pi_i) \neq \epsilon$ iff $f(r,\pi_i) \neq \epsilon$, $i = 1,\ldots,n$. Note that when $\pi = \eta$ and $s = r$, the inference rule comp is an instance of classical resolution. The next lemma shows that the resolution rules for Petri-PDL are sound.

Lemma 54 *The resolution rules for Petri-PDL are sound.*
Proof: Soundness of ires *and* ures (3-13) *follow from soundness of the resolution inference rule for propositional logic (Robinson, 1965). Soundness of* ser1 (3-14) *and* ser2 (3-15) *follow from (PC) and (Du). Soundness of* ref (3-16) *follows from (R_ϵ). Soundness of* comp (3-17) *follows from (PC).* ■

Definition 55 Derivation

A derivation *from a Petri-PDL problem in DSNF_{PPDL} $\mathcal{P} = \langle \mathcal{I},\mathcal{U} \rangle$ by RES_{PPDL} is a sequence $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$ of Petri-PDL problems such that $\mathcal{P}_0 = \mathcal{P}$, $\mathcal{P}_i = \langle \mathcal{I}_i,\mathcal{U}_i \rangle$, and $\mathcal{P}_{i+1}$ is either*

- *$\langle \mathcal{I}_i \cup \mathcal{D},\mathcal{U} \rangle$, where D is the conclusion of an application of* ires*; or*
- *$\langle \mathcal{I}_i,\mathcal{U}_i \cup \mathcal{D} \rangle$, where D is the conclusion of an application of* ures*,* ser1*,* ser2*,* ref*, or* comp*;*

and $D \neq \top$.

We note that the resolvent D is only included in the set of clauses if it is not a tautology. Also, a resolvent is always kept in the simplest form: duplicate literals are removed; $\top$ and $\bot$ are removed from conjunctions and disjunctions with more than one conjunct/disjunct,respectively; conjunctions (resp. disjunctions) with either complementary literals or $\bot$ (resp. $\top$) are simplified to $\bot$ (resp. $\top$).

Definition 56 Refutation

A refutation *for a Petri-PDL problem in* DSNF_{PPDL} $\mathcal{P} = \langle \mathcal{I}, \mathcal{U} \rangle$ *(by* RES_{PPDL}*) is a derivation from* $\mathcal{P}$ *such that for some* $i \geq 0$, $\mathcal{P}_i = \langle \mathcal{I}_i, \mathcal{U}_i \rangle$ *contains a contradiction, where a contradiction is given by either* $\bot \in \mathcal{I}_i$ *or* $\bot \in \mathcal{U}_i$.

A derivation *terminates* if, and only if, either a contradiction is derived or no new clauses can be derived by further application of resolution rules of RES_{PPDL}.

Theorem 57 *The resolution method for Petri-PDL is sound.*
Proof: Soundness of the resolution method for Petri-PDL follows from Lemma 53, which shows that transformation into the normal form is satisfiability preserving, and from Lemma 54, which shows that each of the resolution inference rules is satisfiability preserving. ■

3.6.3 Usage example

Suppose we want to test the formula

$$\begin{aligned}\varphi \;=\; & ([s, \pi_1 \odot \pi_2](p \to q) \to \\ & (([s, \pi_1][f(s, \pi_1), \pi_1 \odot \pi_2]p \wedge [s, \pi_2][f(s, \pi_2), \pi_1 \odot \pi_2]p) \to \\ & ([s, \pi_1][f(s, \pi_1), \pi_1 \odot \pi_2]q \wedge [s, \pi_2][f(s, \pi_2), \pi_1 \odot \pi_2]q)))\end{aligned}$$

for unsatisfiability. Firstly, we transform $\neg\varphi$ into APNF, which results in:

$$\alpha(\varphi) = [s, \pi_1 \odot \pi_2](\neg p \vee q) \wedge \langle s, \pi_1 \odot \pi_2 \rangle p \wedge \langle s, \pi_1 \odot \pi_2 \rangle \neg q$$

The transformation of $\alpha(\varphi)$ into the normal form results in the following clauses:

1.	t_0	$[\mathcal{I}]$	4.	$\neg t_0 \vee \neg[s, \pi_1 \odot \pi_2]\neg p$	$[\mathcal{U}]$
2.	$\neg t_0 \vee [s, \pi_1 \odot \pi_2]t_1$	$[\mathcal{U}]$	5.	$\neg t_0 \vee \neg[s, \pi_1 \odot \pi_2]q$	$[\mathcal{U}]$
3.	$\neg t_1 \vee \neg p \vee q$	$[\mathcal{U}]$			

The refutation proceeds as follows:

6.	$\neg t_0 \vee \neg[s, \pi_1 \odot \pi_2]t_1 \vee \neg[s, \pi_1 \odot \pi_2]\neg q$	$[\mathcal{U}]\mathsf{ser2}, 4, 2$
7.	$\neg t_0 \vee \neg[s, \pi_1 \odot \pi_2]t_1$	$[\mathcal{U}]\mathsf{comp}, 6, 5$
8.	$\neg t_0$	$[\mathcal{U}]\mathsf{comp}, 7, 2$
9.	$\bot$	$[\mathcal{I}]\mathsf{ires}, 8, 1$

3.7 Some examples

In this section it follows some examples of applications.

3.7.1 General applications

Let B be a proposition that means "the markup of the current state includes (b)" and C means "the markup of the current state includes (c)". The token of b by itself in the Petri Net defined in the Figure 3.2 does not implies that v is achieved: $\langle (b), abt_2c\rangle B \rightarrow \neg\langle (c), abt_2c\rangle C$.

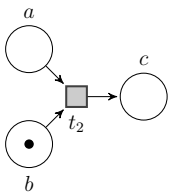


Figure 3.2: A Petri Net where only b has a token

A more abstract example may be composed as: looking at the Petri Net defined in the Figure 3.3, the upper left place (ℓ) is the power button of a vending machine; the bottom left is the coin inserted (m) and the bottom right is the chocolate output (c); if the vending machine is powered on, always when a coin is inserted you will have a chocolate outputed: $\langle (\ell, m), \ell mt_2x \odot xt_3yc \odot yt_1\ell\rangle \top \rightarrow \langle (\ell, m), \ell mt_2x\rangle\langle (x), \ell mt_2x \odot xt_3yc \odot yt_1\ell\rangle \top$.

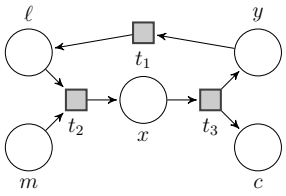


Figure 3.3: A Petri Net for a chocolate vending machine

In the Petri Net example in Figure 2.2, it is possible to say that, in the initial mark, it is not possible that the elevator goes down a floor. That is, $\langle (U, U, U, U), Ut_1D \odot Dt_1U\rangle \top \rightarrow \neg\langle (U, U, U, U, U), Ut_1D \odot Dt_1U\rangle \top$; and that it the elevator goes up a floor it can down too, as in $\langle (U, U, U, U), Ut_1D \odot Dt_1U\rangle \top \rightarrow \langle (U, U, U, D), Ut_1D \odot Dt_1U\rangle \top$.

Concerning the Petri Net example in Figure 2.3, we can say that when a message is being received, it is not possible to receive another at the same time: $\langle (p_1, p_3, p_4), p_1t_3p_1p_2 \odot p_2t_1p_3 \odot p_3p_4t_2p_5 \odot p_5t_1p_4\rangle \top \rightarrow \neg\langle (p_3, p_4), p_1t_3p_1p_2 \odot p_2t_1p_3 \odot p_3p_4t_2p_5 \odot p_5t_1p_4\rangle \top$.

3.7.2
Game modelling and properties verification

This section presents an example of game modelling using Petri Nets and how to verify properties using Petri-PDL. Game formalization that makes use of logical tools to infer properties lacks in some points depending on the kind of the game modelled (Chen et al., 2013).

In the work of de de Oliveira et al. (2011) is presented a formalization to specify and infer properties of a RPG game, the method uses a WorkFlow Petri Nets (WFPN) with linear logic. Although this method is sound it is not complete. Other disadvantage in this formalism involves the restriction of resources used in each transition of the WorkFlow Petri Net. Once the soundness of the formalism is inherited from the soundness of Linear Logic, it lacks on represent the use of a same resource despite of the representation in a Petri Net due to its markup.

One case study in which the Petri Nets can be applied to model is the "Rock-Paper-Scissors" game. It is presented in Figure 3.4 where, once a player inserts a coin in a supposed machine (i.e. the place "Coin" has a token), a transition may fire and the place "Player_1" (the gamer) and "Player_2" (the machine bot) will have a token. Now the player can select one of the options at the same time that the machine can choose one. If the user wins, a token will be placed at "Win_1" and the user will be able to play again; if he loses, at the place "Win_2" or the game restarts if there is a draw match.

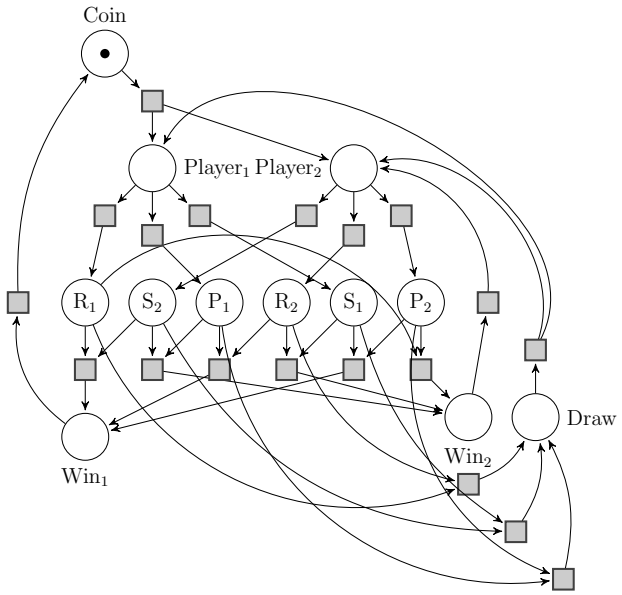
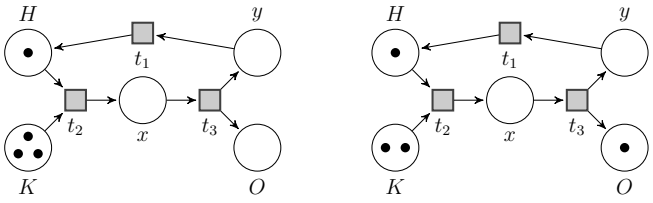


Figure 3.4: Petri Net for "Rock-Paper-Sicissors" game

Consider the Petri Net represented in Figure 3.5(a). The upper left place

(H) is the handle to open a door to the next stage in the actual game level; the bottom left is the key (K) takes by the player in the previous level and the bottom right is the door opening (O); if the key is in the lock, always when the handle is lowered the door for the next stage is open. A token in place x means that the door is unlocked and one in y that the player is handling the door. The formula $\langle(HKKK), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ models this actual state of the Petri Net program in Figure 3.5(a) where after the Petri Net program stops φ will be true in a future state. Supposing that $\langle(HKKK), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ is true, the only enabled transition is t_2, so we have that $\langle(HKKK), HKt_2x\rangle\langle(KKx), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ is true; then t_2 fires and we have that $\langle(KKx), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ is true. Now the only enabled transition is t_3, so we have that $\langle(KKx), xt_3yO\rangle\langle(KKOy), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ will be true and, after the firing of t_3, the formula $\langle(KKOy), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ will be true. Finally t_1 is the only enabled transition. Hence we can now verify that $\langle(KKOy), yt_1H\rangle\langle(KKOH), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ will be true where after the firing of t_1 will achieve the markup of Figure 3.5(b) and $\langle(KKOH), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ will be true. So we can verify that if $\langle(HKKO), HKt_2x \odot xt_3yO \odot yt_1H\rangle\varphi$ is true in an actual state, $\langle(HKKO), HKt_2x \odot xt_3yO \odot yt_1K\rangle\varphi$ is true in a future state (i.e. one door is opened, the player has two keys and it is possible to unlock another door).



3.5(a): Initial markup 3.5(b): After some firings

Figure 3.5: Two markups of a Petri Net for a game where doors separate stages

Note that the player has three keys and, once one of them is used the resource amount is decreased (i.e. one token moves out from place H) and it is possible to set the amount of doors opened (i.e. the amount of tokens in O). There is, it is possible to deal with count of resources and its consumption.

Another property inherited from Petri Nets is the simplified way to deal with concurrence. In a scenario where the player cannot only open a door (as in Figure 3.5(a)) but there is also the possibility to take an object disposed into the room, as in Figure 3.6, where place B denotes that the user is taking same object in the current room. If t_1' fires then t_2 will be unable to fire. So, it is possible to model in Petri-PDL that when a player is taking something in the current room he cannot open a door. So, modelling the scenario in Figure 3.6

in a formula ψ we have that $\psi = \langle(KKK, B), HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H\rangle\top \rightarrow \langle(KKK, B), HKt_2x\rangle\langle\epsilon, HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H\rangle A$, where A is some property true after the running of this program. Looking to the left side of the implication, as t''_1 is able to fire it is straightforward that the left side of the implication is true by the semantical notion of satisfaction of Petri-PDL. But, for the right side of the implication, t_1 is not able to fire, so cannot be true, hence its negation is true and the formula is valid. Note that the ϵ in the second modality of the right side of the implication is the result of the firing function for $f(KKKB, HKt_2x)$. Using the Resolution system for Petri-PDL to verify if this formula holds we have that $\alpha(\psi) = \langle(KKKB), HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H\rangle A \wedge [(KKK, B), HKt_2x][\epsilon, HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H]\neg A$ (i.e. ψ in the APNF). The transformation of $\alpha(\psi)$ into the normal form results in the following clauses:

1.	t_0	$\mathcal{I}$
2.	$\neg t_0 \vee \neg[(KKKB), HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H]\neg A$	$[\mathcal{U}]$
3.	$\neg t_0 \vee [(KKK, B), HKt_2x]t_1$	$[\mathcal{U}]$
4.	$\neg t_1 \vee [\epsilon, HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H]t_2$	$[\mathcal{U}]$
5.	$\neg t_2 \vee \neg A$	$[\mathcal{U}]$

The refutation proceeds as follows:

6.	$\neg t_0 \vee \neg t_1 \vee [\epsilon, HKt_2x \odot xt_3yO \odot yt_1H \odot Ht'_1B \odot Bt''_1H]\neg A$	$[\mathcal{U}, \mathsf{ser2}, 4, 5]$
7.	$\neg t_0 \vee \neg t_1$	$[\mathcal{U}, \mathsf{comp}, 6, 2]$
8.	$\neg t_0 \vee t_1$	$[\mathcal{U}, \mathsf{ref}, 3]$
9.	$\neg t_0$	$[\mathcal{U}, \mathsf{ures}, 7, 8]$
10.	$\bot$	$[\mathcal{I}, \mathsf{ires}, 1, 9]$

The same methodology may be applied to many board games.

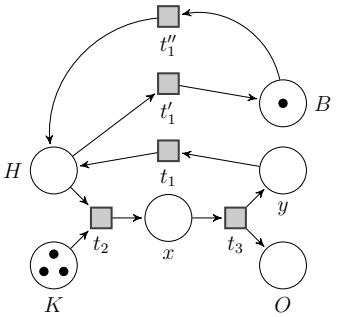


Figure 3.6: A Petri Net for a game where the user open doors with keys

3.8
Petri Nets modelling limitations

As will be presented in the next chapter, ordinary Petri Nets are not able to deal with many situations involving time. That is no way to model a scenario in which some transition fires necessarily more than others. To increase the expressiveness of the logic, we will introduce an extension of Petri Nets (Stochastic Petri Nets) and present an extension of Petri-PDL to deal with this kind of Petri Nets.