Títol: *"Una eina per verificar propietats en xarxes de Petri usant àlgebra lineal"*

Volum: *1/1*

Alumne: *Sara Royuela Alcázar*

Director/Ponent: *Josep Carmona Vargas*

Departament: *Llenguatges i Sistemes Informàtics*

Data: *1 de Juliol de 2010*

DADES DEL PROJECTE

Títol del Projecte:	*"Una eina per verificar propietats en xarxes de petri usant àlgebra lineal"*
Nom de l'estudiant:	*Sara Royuela Alcázar*
Titulació:	*Enginyeria Superior en Informàtica*
Crèdits:	*37,5*
Director/Ponent:	*Josep Carmona Vargas*
Departament:	*Llenguatges i Sistemes Informàtics*

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President:	*Jordi Cortadella Fortuny*
Vocal:	*Vera Sacristán Adinolfi*
Secretari:	*Josep Carmona Vargas*

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Agraïments

A la mama, al papa i al tete, el més important.

Als diumenges com Déu mana, amb tu.

A la pluja dels dilluns i al sol dels dimecres, als meus nens.

Al meu PC, que ha aguantat com un campió.

A tants anys de dedicació.

Índex

1 Introducció

En aquest capítol s'introdueix una vista general del projecte i s'explica la utilitat, tant de verificar propietats en Xarxes de Petri, com d'usar la Programació Lineal per fer-ho.

1.1 Context del projecte

Els sistemes que ens envolten, tant els que no depenen directament dels humans (els processos naturals) com els que sí (els processos industrials) són complexos. L'evolució de la societat i de les tecnologies requereix del coneixement acurat del comportament d'aquests sistemes. Igualment, d'aquells sistemes que ara només són idees i es volen implantar.

Una via per analitzar-los i ajudar en la millora dels mateixos sense modificar el nostre món és l'ús de **models**. Si aquests models representen detalladament els sistemes reals, de manera que es minimitzi el risc de trobar imprevistos, haurem trobat una bona base per analitzar les propietats de la realitat representada i, a més, un mode de portar els sistemes a situacions límit molt menys costosa que fer-ho en àmbits reals.

Existeixen molts models de comportament dinàmic que ens permeten representar la realitat dels sistemes. En concret, les **Xarxes de Petri** (en endavant **XdP**) són un model de representació de sistemes distribuïts caracteritzats per ser concurrents, asíncrons, paral·lels, no deterministes i/o estocàstics.

Per abordar l'anàlisi de sistemes mitjançant un model es pot fer diferent tipus de proves, com per exemple tests de càrrega, simulació d'estats límit o verificació de propietats. En aquest projecte es descriu una metodologia per a la verificació de propietats en sistemes modelats mitjançant XdP.

D'altra banda, és necessari que la complexitat del procés d'anàlisi dels models es minimitzi. Moltes de les propietats verificables sobre els sistemes tenen un cost NP-complet, però utilitzant diferents mètodes d'anàlisi que es veuran més endavant en aquest document, extraurem sistemes d'inequacions que representin les propietats per utilitzar així mètodes d'aproximació ens aportin un bon balanç entre fiabilitat i temps de computació. Dins d'aquest marc, la **Programació Lineal** (en endavant **PL**), un mètode matemàtic que, mitjançant equacions lineals, permet optimitzar (maximitzar o minimitzar) una funció objectiu, també lineal, és un mètode vàlid, el que s'ha aplicat en aquest projecte, per verificar les propietats definides sobre la XdP.

1.2 Motivació i objectius del projecte

L'**objectiu** d'aquest projecte és la creació d'una eina que permeti verificar propietats simples en sistemes modelats amb XdP basada en l'àlgebra lineal. Per això caldrà fer un estudi previ del funcionament de les XdP i de l'estat de l'art de l'aplicació de l'àlgebra lineal en aquest tipus de models.

L'interès d'aplicar l'àlgebra lineal sobre les XdP ve **motivat** pel fet de ser un model senzill i manejable; en canvi altres representacions del mateix tipus de sistemes com l'arbre d'abastabilitat i el graf d'abastabilitat poden arribar a tenir un cost exponencial respecte dels elements del sistema. En aquests casos el temps de càlcul creix notablement i això fa més difícil l'anàlisi.

Podem agrupar els objectius del projecte de la següent manera:

1. Xarxa de Petri
   - 1.1. Estudi de les XdP: representació i comportament.
   - 1.2. Definició de les propietats que es vol verificar.
   - 1.3. Detecció de la informació necessària d'una XdP per a l'anàlisi de les propietats del sistema que modelitza.
   - 1.4. Anàlisi sintàctica de XdP a partir del format d'entrada escollit.
   - 1.5. Creació d'una estructura de dades per guardar la informació rellevant per a l'anàlisi de propietats da la XdP.
2. Programació lineal
   - 2.1. Definició de l'àlgebra (model de PL) que emmagatzema les propietats a validar sobre la XdP.
   - 2.2. Ús d'un solucionador de problemes basat en PL per resoldre el problema.
3. Propietats
   - 3.1. Definició d'una gramàtica que permet comprovar la validesa de les propietats que es vol verificar.
   - 3.2. Creació d'un estructura de dades que contingui les propietats i la seva relació a partir del recorregut de l'arbre creat amb l'anàlisi gramatical.
   - 3.3. Transformació d'aquestes dades en equacions de programació lineal comprensibles pel solucionador.
   - 3.4. Verificació de les propietats amb un solucionador de problemes de PL.
4. Presentació a l'usuari
   - 4.1. Creació d'una interfície gràfica per explotar l'eina descrita.
   - 4.2. Creació d'una llibreria amb les funcionalitats descrites.
5. Creació de la documentació necessària sobre el projecte.

1.3 Estructura del document

S'ha estructurat aquest document en capítols de la següent manera:

1. **Introducció**
   Descripció de les motivacions que m'han portat a realitzar aquest projecte juntament amb els objectius que s'ha plantejat per a la seva realització.
2. **Teoria sobre XdP i PL**
   Introducció al món de les XdP, definició de les característiques i de les propietats que són rellevants en aquest projecte, així com diferents tècniques d'anàlisi.
   Descripció de conceptes bàsics sobre PL.
   Resum de l'estat de l'art en l'aplicació de la PL en XdP.
   Descripció de les propietats que són acceptades per l'eina desenvolupada.
3. **Especificació**
   Anàlisi dels requisits: definició de l'abast del sistema desenvolupat per concretar què ha de fer el sistema i quins objectius específics cal assolir. S'ha separat aquest anàlisi en requisits funcionals i no funcionals.
   Anàlisi funcional: definició del model de dades mitjançant la representació dels conceptes (objectes) més significatius, introducció dels actors i els casos d'ús del sistema.
4. **Disseny**
   Definició de l'arquitectura del sistema, concreció del model de dades, mitjançant el diagrama de classes, i del comportament d'aquestes dades, mitjançant diagrames de seqüència.
5. **Implementació**
   Descripció de les tecnologies utilitzades durant el desenvolupament de l'aplicació i de la implementació realitzada, valorant les diferents alternatives i justificant les decisions preses.
6. **Resultats i exemples**
   Diferents exemples amb els resultats obtinguts i la seva valoració.
7. **Planificació i Anàlisi econòmica**
   Planificació temporal per a la realització del projecte i anàlisi econòmica del mateix.
8. **Conclusions**
   Exposició de les valoracions que he extret del projecte i descripció de les possibles ampliacions del mateix.
9. **Annexes**
   Extensions sobre la teoria utilitzada al projecte aplicable en futures ampliacions de l'eina.

2 Teoria

La informació que s'inclou aquest apartat té la finalitat d'introduir al lector en els aspectes teòrics aplicats en l'eina desenvolupada.

2.1 Xarxes de Petri

Les XdP o **Xarxes de Lloc/Transició** són una generalització de la teoria d'autòmats, definida l'any 1962 per Carl Adam Petri al seu treball doctoral "*Kommunikation mit Automaten*", que defineix una representació matemàtica i gràfica de sistemes distribuïts.

Moltes dades incloses en aquest punt provenen de l'article "*Petri Nets: Properties, Analysis and Applications*" de Tadao Murata (1).

A continuació es defineix formalment una XdP i s'explica, d'una banda, les propietats sobre les XdP, i d'altra, els mètodes d'anàlisi sobre XdP que s'apliquen respectivament en l'ús i el desenvolupament de l'eina. A l' "Annex 1" es pot trobar una ampliació tant sobre propietats, com sobre mètodes d'anàlisi, aplicables a futures ampliacions de que permetran una anàlisi més acurada.

2.1.1 Definició bàsica

Una XdP és un graf dirigit bipartit format per:

- **Nodes**, que es representen de dos maneres segons el seu significat:
  - **Transicions**: representa esdeveniments i es simbolitza amb barres.
  - **Llocs**: representa condicions i es simbolitza amb cercles. Contenen un nombre natural de **marques** o **tokens**, i la distribució d'aquestes marques sobre els llocs de la XdP s'anomena **marcatge**. (La XdP pot tenir un marcatge inicial, anomenat M_0).
- **Arcs dirigits**: representen les relacions que hi ha entre les condicions i els esdeveniments i es simbolitzen amb fletxes. Un arc des d'un lloc *A* a una transició *B* descriu que *A* és pre-condició de *B*, i un arc entre *B* i un lloc *C* descriu que *C* és post-condició de *B*. Els arcs poden tenir associat un pes.

Formalment, podem dir que una XdP és una 5-tupla, $PN = (P, T, F, W, M_0)$ on:

- $P = (p_1, p_2, \dots, p_n)$ és un conjunt finit de llocs,
- $T = (t_1, t_2, \dots, t_m)$ és un conjunt finit de transicions,
- $F \subseteq (P \times T) \cup (T \times P)$ és un conjunt d'arcs (relació de flux),
- $W: F \rightarrow \{1, 2, 3, \dots\}$ és una funció de pes,
- $M_0: P \rightarrow \{0, 1, 2, 3, \dots\}$ és el marcatge inicial i
- $P \cap T = \varnothing$ i $P \cup T \neq \varnothing$.

Una XdP sense marcatge inicial es defineix com $N = (P, T, F, W)$. Una XdP amb marcatge inicial i, simplificant la notació inicial, es defineix com $(N, M0)$. En aquest projecte es treballa amb xarxes marcades.

Definim els llocs i transicions d'entrada i de sortida com **pre-conjunt** i **post-conjunt** de la següent manera:

- ${}^\bullet t = \{p \mid (p, t) \in F\}$ = pre-conjunt de t = conjunt de llocs d'entrada de t.
- $t^\bullet = \{p \mid (t, p) \in F\}$ = post-conjunt de t = conjunt de llocs de sortida de t.
- ${}^\bullet p = \{t \mid (t, p) \in F\}$ = pre-conjunt de p = conjunt de transicions d'entrada de p.
- $p^\bullet = \{t \mid (p, t) \in F\}$ = post-conjunt de p = conjunt de transicions de sortida de t.

Es pot definir el comportament d'una XdP en funció dels seus estats. L'estat (o marcatge) d'una XdP canvia segons les següents normes de transició:

- Una transició t està **activa** si per a cada lloc p d'entrada a t, p està marcat amb, al menys, $w(p,t)$ marques (pes de l'arc que va des de p fins a t).
- Una transició activa es pot disparar o no.
- El dispar d'una transició activa elimina $w(p,t)$ marques de cada lloc p d'entrada a la transició t i afegeix $w(t,p)$ marques a cada lloc p de sortida de t.

A la següent imatge es mostra un exemple simple del funcionament d'una XdP que modelitza la creació d'aigua a partir dels seus elements, hidrogen i oxigen:

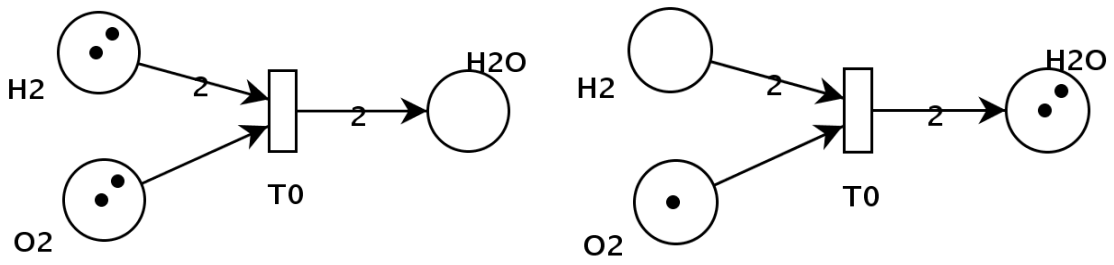


Figura 1: a l'esquerra, marcatge abans de disparar la transició activa *T0*; a la dreta, marcatge després de disparar la transició *T0*, ara desactivada.

2.1.2 Comportament a les Xarxes de Petri

En aquest apartat es mostra:

- Un conjunt d'estructures d'interès.
- Un conjunt de propietats, algunes basades en les anteriors estructures d'interès.

Aquestes qualitats són molt útils en l'estudi de les XdP ja que permeten conèixer alguns comportaments molt rellevants per l'anàlisi dels sistemes que modelen.

2.1.2.1 Estructures d'interès

2.1.2.1.1 *Paral·lelisme i concurrència*

Dues transicions són **concurrents** si són causalment independents, és a dir, una transició pot disparar-se abans, després o paral·lelament a l'altra.

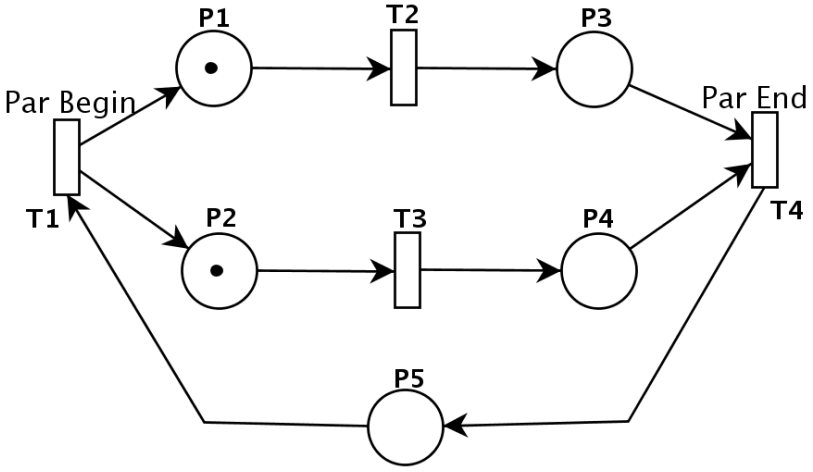


Figura 2: XdP presentant activitat paral·lela
[concurrència entre les transicions T2 i T3]

2.1.2.1.2 *Conflicte*

Es defineix com **conflicte** o decisió l'estat en què un lloc *p* té dos o més transicions de sortida.

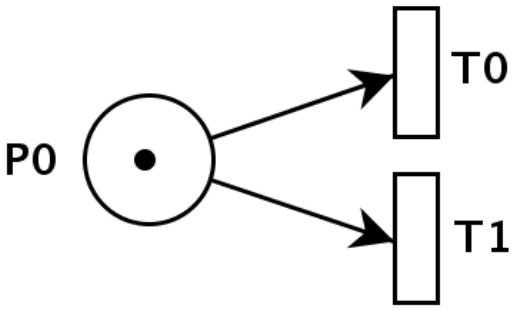


Figura 3: Situació de conflicte en una XdP

2.1.2.1.3 *Confusió*

S'anomena **confusió** en una XdP a la situació en què es dóna concurrència i conflicte.

Podem distingir dos tipus de confusió:

- **Simètrica**: dues transicions t_1 i t_2 són concurrents i estan ambdues en conflicte amb la mateixa transició t_3.
- **Asimètrica**: dos transicions t_1 i t_2 són concurrents i poden estar en conflicte amb una altra transició t_3 només si es compleix alguna seqüència de dispars determinada.

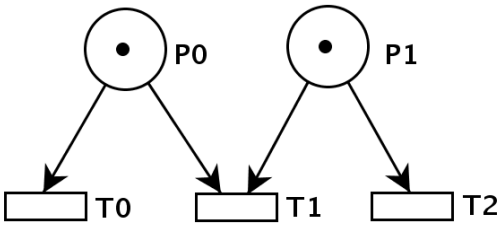


Figura 4: Confusió simètrica:
T0 i *T2* són concurrents i estan en conflicte amb *T1*

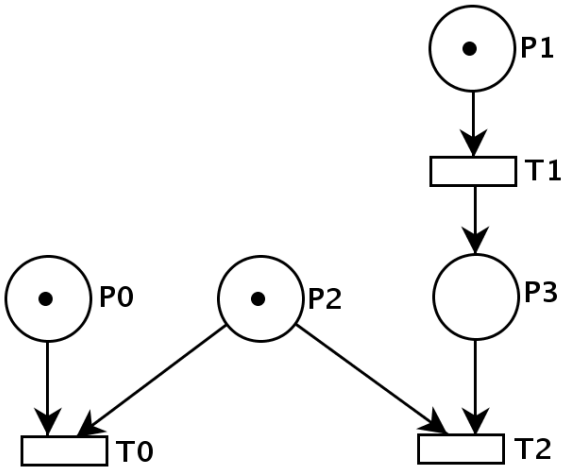


Figura 5: Confusió asimètrica:
T0 és concurrent amb *T1* i estarà en conflicte amb *T2* si *T1* es dispara abans que *T0*

2.1.2.2 *Propietats*

Les propietats de les XdP permeten detectar fenòmens d'interès o errors de funcionament en el sistema que modelen.

Es pot distingir dos tipus diferents de propietats:

- **Propietats comportamentals**: depenen del marcatge inicial.
- **Propietats estructurals**: independents del marcatge inicial.

A continuació es defineix algunes de les propietats comportamentals més importants i que són les que l'eina desenvolupada permet verificar (abastabilitat, ciclicitat, conflictivitat i exclusió mútua) indicant quina és la informació que aporta cadascuna sobre el model representat per la XdP. A l'annex 2 , i es fa una breu explicació de les propietats estructurals per tenir-ne una petita noció.

2.1.2.2.1 Propietats comportamentals

2.1.2.2.1.1 *Abastabilitat*

Definició

Un marcatge M_n és **abastable** des d'un marcatge M_0 si existeix una seqüència de dispars que transforma M_0 en M_n.

Anomenarem $\sigma = t_1\ t_2\ ...\ t_m$ a qualsevol seqüència de dispars sobre una XdP.

S'anomena $R(N, M_0)$ o $R(M_0)$ al conjunt de marcatges abastables a N des de M_0.

S'anomena $L(N, M_0)$ o $L(M_0)$ al conjunt de seqüències de dispars a N des de M_0.

Característiques

A partir d'aquesta propietat es pot definir el **problema de l'abastabilitat**, que consisteix a trobar si $M_n \in R(M_0)$. Aquest és un problema decidible, però exponencial en temps i espai per a casos generals.

2.1.2.2.1.2 *Ciclicitat*

Definició

Una XdP té un **comportament globalment cíclic** per un marcatge inicial M_0 si existeix una seqüència de dispars que permet assolir el marcatge inicial M_0 a partir de qualsevol marcatge M_i successor de M_0.

Si relaxem aquesta propietat, podem definir un **estat "casa"** M' si per a cada marcatge M de $R(M_0)$, si M' és abastable a partir de M.

Característiques

La ciclicitat d'una XdP per un marcatge M_0 garanteix la no existència de subconjunts d'**estats finals** (conjunts d'estats a partir dels quals no són abastables altres estats successors de M_0).

Exemples

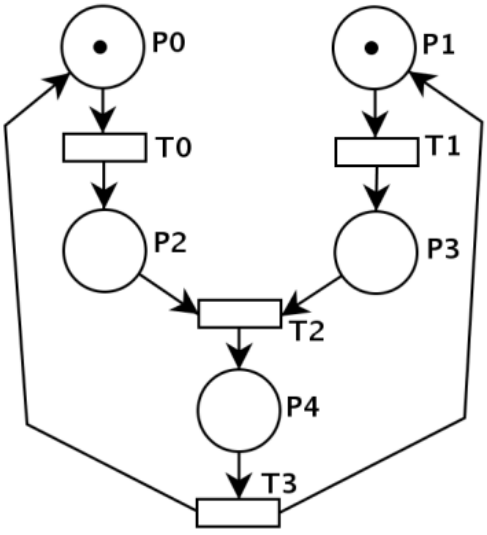


Figura 6: XdP amb comportament global cíclic

2.1.2.2.1.3 *Conflictivitat*

Definició

En una XdP existeix un **conflicte estructural** quan un lloc té més d'una transició de sortida.

Dos transicions t_i i t_j estan en **conflicte efectiu** per M_0 si:

- existeix un marcatge abastable des de M_0 que activa simultàniament a t_i i t_j i
- al disparar-se t_i (t_j) el marcatge que s'obté no activa t_j (t_i).

Un conjunt de transicions $T_C \subset T$ està **en conflicte** si el dispar de qualsevol subconjunt de transicions $\{t_i\} \subset T_C$ dona com a resultat un marcatge en el qual alguna transició $t_i \subset T_C$ es desactiva.

Una XdP és **persistent** si no existeix un conjunt de transicions que estigui en conflicte per a cap marcatge M del conjunt d'abastabilitat.

Característiques

La situació de conflicte efectiu és normalment inacceptable per a qualsevol descripció d'un sistema, ja que es correspon amb una situació ambigua. Aquest conflicte es resol mitjançant una interpretació associada a la xarxa.

Exemples

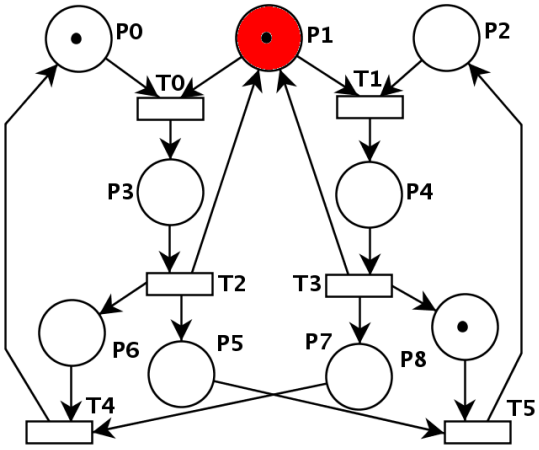


Figura 7: XdP amb conflicte estructural

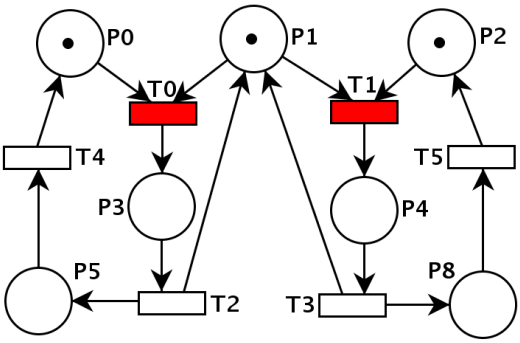


Figura 8: XdP amb conflicte efectiu

2.1.2.2.1.4 *Exclusió mútua*

Definició

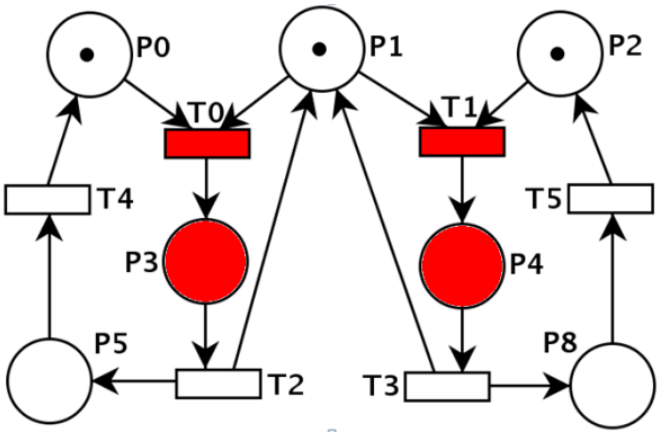
Un conjunt de transicions $T_e \subset T$ és **mútuament exclusiu** si el dispar de qualsevol transició $t_i \subset T_e$ genera un marcatge en el qual totes les transicions $t_j \subset T_e$ on $i \neq j$ es desactiven.

Dos llocs d'una XdP estan en **exclusió mútua** per un marcatge M_0 si no poden estar marcades simultàniament a cap marcatge abastable des de M_0.

Característiques

L'exclusió mútua serveix per modelar l'ús de recursos compartits per múltiples usuaris.

Exemples



Els llocs P3 i P4, i les transicions T0 i T1 són mútuament exclusius.

- P1, recurs disponible
- P3, recurs pres per T0
- P4, recurs pres per T1

Figura 9: Exclusió mútua

2.1.3 Mètodes d'anàlisi

Gràcies a que les XdP tenen una gran potència de modelat, l'anàlisi del seu comportament pot donar molta informació sobre el sistema que representen. Les principals tècniques d'anàlisi de les XdP són:

- **Tècniques enumeratives**: basades en l'arbre d'abastabilitat.
- **Tècniques estructurals**: basades en equacions matricials o d'estat.
- **Tècniques de transformació**: basades en la reducció o la descomposició.

En l'eina desenvolupada només s'utilitzen tècniques estructurals, però es considera interessant fer una introducció, principalment, a les tècniques enumeratives, ja que permeten obtenir solucions sempre vàlides, tot i que tenen les seves limitacions, i aquest és el motiu principal de l'elecció de les tècniques estructurals. A l' "Annex 1" es pot trobar una petita ampliació de les tècniques enumeratives i una introducció a les tècniques de transformació.

2.1.3.1 Tècniques enumeratives

L'**arbre d'abastabilitat** és la representació en forma d'arbre del conjunt de marcatges abastable en una XdP, $R(M)$, a partir d'un marcatge inicial M_0.

Per mantenir un arbre d'abastabilitat finit en el cas que esdevingui infinit, s'introdueix el símbol ω que es pot veure com un infinit. Té les propietats que:

- $\omega > n$,
- $\omega \pm n = \omega$,
- $\omega \geq \omega$.

Una altra manera de representar l'evolució dinàmica d'un model en xarxa controlat per marcatges com és el cas de les XdP és el graf d'abastabilitat. El **graf d'abastabilitat** d'una XdP N és un graf dirigit etiquetat en què els nodes són els estats de N i els arcs, etiquetats

amb els esdeveniments (transicions) de *N*, representen un canvi de configuració degut al dispar d'una transició.

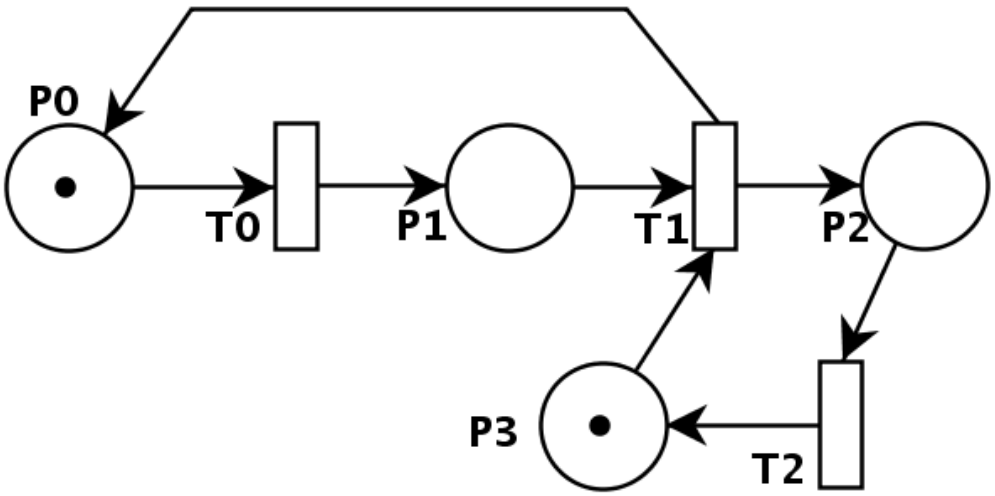


Figura 10: XdP limitada (l'arbre d'abastabilitat -Figura 11- no conté ω)

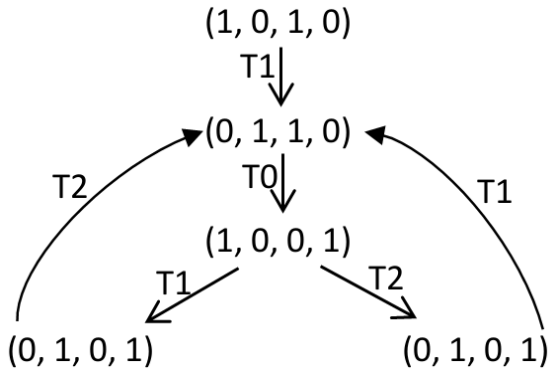


Figura 11: Arbre d'abastabilitat de la XdP de la Figura 10

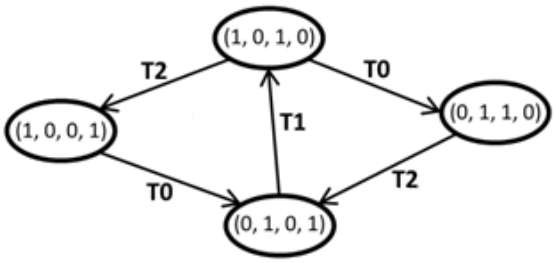


Figura 12: Graf d'abastabilitat de la XdP de la Figura 10

Aquestes tècniques són aplicables en teoria, però a la pràctica es veuen limitades a sistemes petits degut a l'elevada complexitat computacional, ja que l'arbre i el graf d'abastabilitat pot esdevenir complex, gran i immanejable, fet que s'anomena **explosió d'estats**.

2.1.3.2 Tècniques estructurals

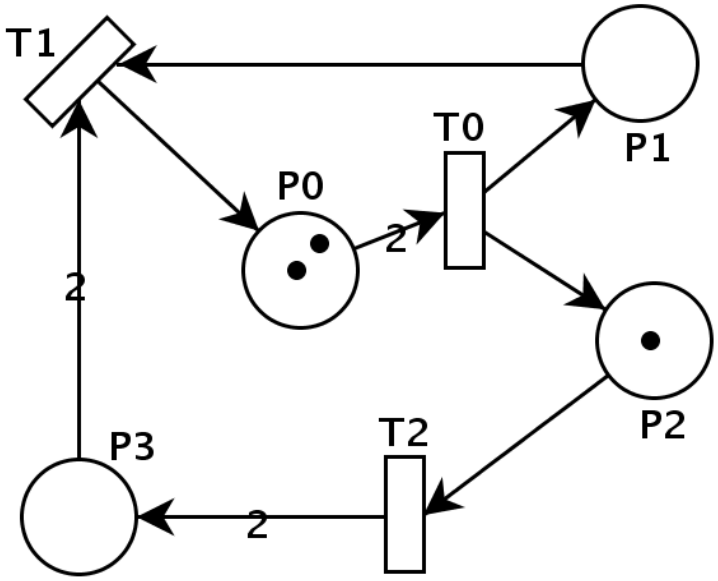
El conjunt de tècniques estructurals es basa en l'obtenció de la informació que hi ha al model en funció de la seva estructura i del marcatge inicial. El comportament dinàmic dels sistemes modelats amb XdP es descriu transformant aquesta informació a un conjunt d'equacions algebraiques.

2.1.3.2.1 Anàlisi matricial

2.1.3.2.1.1 *Matriu d'incidència*

Per una XdP *N* amb *n* llocs i *m* transicions, la **matriu d'incidència** *A = [aij]* és una matriu de *n x m* enters i el valor de l'element $aij = aij^{+} - aij^{-}$, on $aij^{+} = w(i, j)$ és el pes de l'arc que va des del lloc *i* a la seva transició de sortida *j*, i $aij^{-} = w(j, i)$ és el pes de l'arc que va des de la transició *j* al seu lloc *i* de sortida.

Podem utilitzar la teoria matricial per analitzar propietats del comportament dinàmic com per exemple: "Una transició *t* està activa en un marcatge *M* si i només si $aij^{-} \leq M(j)$, on *j=1,2,...,m*".



$$\begin{pmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{pmatrix}$$

Figura 13: XdP amb la seva matriu d'incidència

2.1.3.2.1.2 *Equació de marcatge*

Definim el vector de fila M_k com un vector de *n* nombres on la i-èssima posició denota el nombre de marques que hi ha al lloc *i* immediatament després del k-èssim dispar d'alguna seqüència determinada de dispars. El k-èssim **vector de dispars** u_k és un vector de fila de m – 1 0's i un 1 a la posició que representa que la transició *j* es dispara al k-èssim dispar.

S'anomena **equació de marcatge** l'equació que dóna la transició necessària per que el marcatge M_k sigui abastable des del marcatge M_{k-1}. S'escriu com $M_k = M_{k-1} + A \cdot u_k$, per a $k = 1, 2, ...$

Condició d'abastabilitat

M_d és un marcatge abastable des de M_0 amb una seqüència de dispars $\{u_1, u_2, ..., u_d\}$ si es compleix l'equació $M_d = M_0 + A \cdot \sum_{k=1}^{d} u_k$. També podem escriure $A \cdot x = \Delta M$ on $\Delta M = M_d - M_0$ i $x = \sum_{k=1}^{d} u_k$ on *x* és un vector de fila *m* enters positius anomenat **vector de comptatge de dispars**. La j-èssima entrada d'aquest vector indica el nombre de vegades que s'ha de disparar la transició j per transformar M_0 en M_d.

A continuació es mostra una XdP amb dos exemples d'aplicació de l'equació de marcatge, i com un marcatge M_d és abastable i l'altre no:

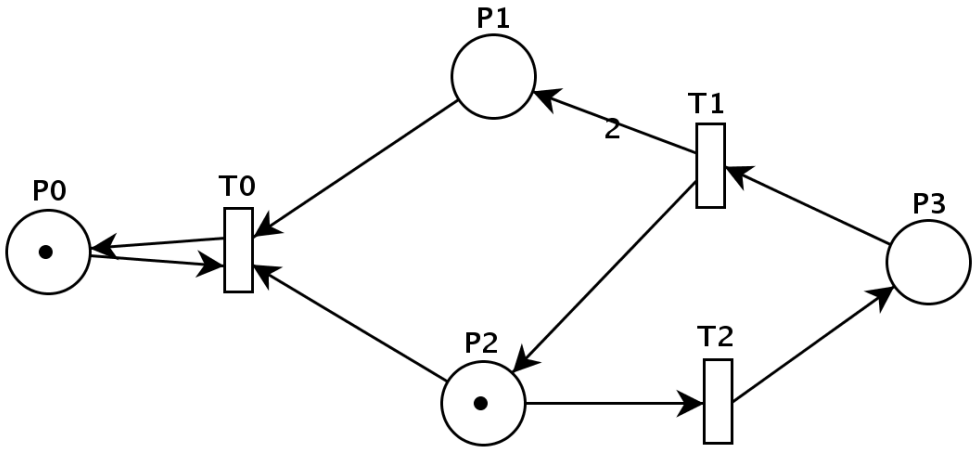


Figura 14: Xarxa de Petri

Marcatge abastable	**Marcatge no abastable**
Marcatge inicial: $M_0 = (1,0,1,0)$ Marcatge final: $M_d = (1,3,0,0)$ Equació de marcatge: $[1 \quad 3 \quad 0 \quad 0] - [1 \quad 0 \quad 1 \quad 0] = \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \cdot x$ Solució per l'equació de marcatge: - Seqüència de dispars: $\sigma = t_3, t_2, t_3, t_2, t_1$ - Vector de comptatge de dispars: $x = (1,2,2)$ L'equació de marcatge té solució, per tant, el marcatge $M_d = (1,3,0,0)$ és abastable des de $M_0 = (1,0,1,0)$.	Marcatge inicial: $M_0 = (1,0,1,0)$ Marcatge final: $M_d = (1,7,0,1)$ Equació de marcatge: $[1 \quad 7 \quad 0 \quad 1] - [1 \quad 0 \quad 1 \quad 0] = \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \cdot x$ No existeix solució per l'equació de marcatge, per tant, el marcatge $M_d = (1,7,0,1)$ no és abastable des de $M_0 = (1,0,1,0)$.

2.1.3.2.1.3 *Problemes associats a l'anàlisi matricial*

Tot i que l'anàlisi de XdP mitjançant tècniques matricials és molt prometedor, també presenta problemes que ens poden portar a buscar altres formes d'enriquir les restriccions que han de complir les solucions de l'equació de marcatge. Algunes d'aquestes limitacions són:

- La matriu d'incidència no reflexa totalment l'estructura de la xarxa. Per exemple, les transicions que tenen arcs d'entrada i de sortida cap a un mateix lloc (auto-bucles) es cancel·len a la matriu d'incidència.
- El vector de comptatge de dispars no dóna informació sobre l'ordre en que s'executa els dispars, només de quines transicions es disparen i quantes vegades ho fan.
- L'existència de solució per a l'equació de marcatge és una **condició necessària però no suficient** per que el marcatge sigui abastable, ja que pot ocórrer que la solució trobada no es correspongui amb cap seqüència de dispars possible.

Aquestes limitacions ens portaran a encetar un camí addicional a l'ús de l'equació de marcatge per enriquir l'eina desenvolupada.

2.1.3.2.2 Invariants

La informació d'aquest punt s'extreu de l'apartat de "S-invariants i T-invariants" (3) del llibre ja mencionat "*Free Choice Petri Nets*" i de l'article "*Basic Linear Algebraic Techniques for Place/Transition Nets*", de Jörg Desel (3).

Anomenen **invariant** d'un sistema dinàmic a una asserció que es manté a tots els estats abastables del sistema.

En sistemes representats per una xarxa, com les XdP, és possible extreure vectors de nombres racionals que són invariants al sistema. A continuació veurem dos vectors invariants anomenats **S-invariants** i **T-invariants**.

Aquests vectors es poden utilitzar per a la verificació de propietats comportamentals. Es considera important parlar a l' "Annex 1" d'aquestes aplicacions ja que, tot i que l'eina permet verificar si un vector és un invariant, no permet cercar tots els invariants d'una XdP, tasca que permetria utilitzar-los com a mètode de verificació. Es proposa fermament com a ampliació de l'aplicació.

2.1.3.2.2.1 *S-invariants*

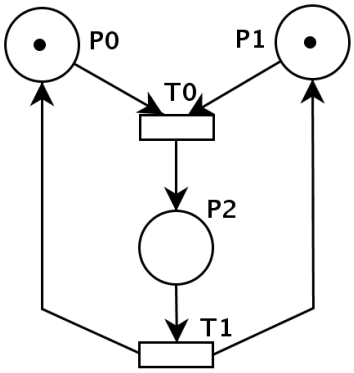
Un **S-invariants** és un vector que conté un conjunt de llocs pels quals el nombre total de marques no varia després de disparar qualsevol combinació factible de transicions.

Donada una xarxa que representa un sistema arbitrari, és complex caracteritzar tots els vectors tals que el seu producte amb la matriu d'incidència sigui constant per a tots els marcatges abastables, no obstant això, és senzill derivar una condició suficient a partir de l'equació de marcatge.

Un **S-invariants**, també anomenat P-invariant, d'una XdP, on A és la matriu d'incidència, és un vector x de m elements racionals que té solució per l'equació $xA = 0$.

Alternativament, podem dir que el mapeig $I: P \rightarrow Q$ és un **S-invariants** si per cada transició t d'una XdP es compleix que $\sum_{p \in {}^\circ t} I(p) = \sum_{p \in t^\circ} I(p)$.

A continuació es mostra una XdP i l'espai de S-invariants per la mateixa:



$$A = \begin{pmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}$$

x = { (1 0 1), (0 1 1) }

Figura 15: XdP amb la seva matriu d'incidència i l'espai de S-invariants

2.1.3.2.2.2 *T-invariants*

Un **T-invariant** és un vector que conté una seqüència de dispars que dóna com a marcatge resultant el marcatge inicial.

Un **T-invariant** d'una XdP, on A és la matriu d'incidència, és un vector y d'elements racionals que té solució per l'equació $Ay = 0$.

Alternativament, podem dir que el mapeig $J: T \rightarrow Q$ és un **T-invariant** si per cada transició p d'una XdP es compleix que $\sum_{t \in {}^\circ p} J(t) = \sum_{t \in p^\circ} J(t)$.

A continuació es mostra una XdP i el resultat d'executar la seqüència de dispars indicada per un T-invariant:

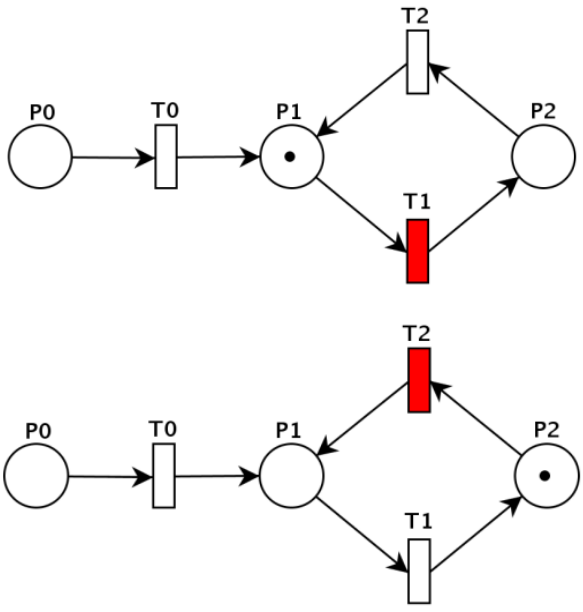


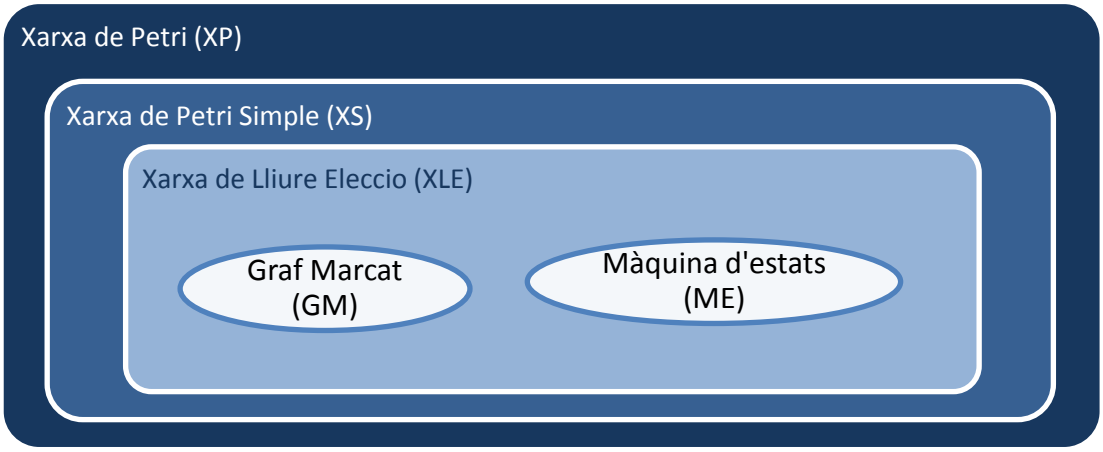
Figura 16: Estats d'una XdP e executar la seqüència de dispars indicada pel T-invariant y=(0 1 1)

2.1.4 Subclasses de Xarxes de Petri

Tot i que les XdP es basen en regles bastant simples, poden tenir comportaments molt complicats. És per això que s'ha definit classes restringides de XdP que permetin modelar un ampli ventall de sistemes i que, per als problemes de més interès, es puguin analitzar amb procediments més o menys simples.

El primer biaix que em de fer és restringir el pes dels arcs de la XdP. Definim una XdP com **ordinària** si tots els seus arcs tenen pes igual a 1. Aquest és el tipus de xarxes que accepta l'eina.

Al següent gràfic es pot observar l'estructura que caracteritza les subclasses de XdP:



A l' "Annex 1" es pot trobar la definició dels subtipus i les seves característiques principals. Únicament fer notar que, degut a les restriccions de la seva definició, tot el grup de xarxes que es troba dins de les XLE té un cost de càlcul molt menor.

2.2 Programació Lineal

En aquest punt s'introdueix superficialment els conceptes bàsics sobre la PL. La informació que s'inclou està extreta principalment de la web de la Wikipèdia: http://en.wikipedia.org/wiki/Linear_programming (5).

S'anomena **Programació Lineal** al conjunt de tècniques matemàtiques que pretenen resoldre problemes en què s'hagi d'optimitzar (maximitzar o minimitzar) un funció objectiu lineal d'una o més variables a partir d'una sèrie de restriccions que poden ser expressades mitjançant inequacions lineals.

Un problema de PL està format per tres elements bàsics:

- **Variables**: nombres reals que han de ser més grans o iguals que zero, $X_i \geq 0$.
- **Restriccions**: podem tenir tres tipus de restriccions, segons l'operador, $A_j \ \ OP \ \ \sum_{i=1}^{N} a_{i,j} \times X_i$ on,
  - A_j, és el valor que caldrà respectar depenent del valor de *OP*,
  - *OP*, és l'operador d'igualtat, que pot prendre els valors =, ≤ o ≥,

- j, és el número de l'equació, que pot anar de 1 a M (nombre total de restriccions),
- i, és el número de la incògnita, que pot anar de 1 a N,
- a, és un coeficient tècnic conegut
- X, són les incògnites, que van de 1 a N,

- Funció objectiu: funció que es vol maximitzar o minimitzar i que té la forma $\sum_{i=1}^{N} f_i \times X_i$.

És important veure la diferencia entre dues formes d'aplicació de la PL:

- **PL** o **Programació Lineal Real**

  Les variables del problema estan al domini dels Reals.

  Existeixen mètodes d'optimització com **Simplex**, un algoritme iteratiu que, mitjançant el seqüenciament d'iteracions, s'aproxima a una solució òptima, en cas d'existir, per a un problema de PL. La seva complexitat és polinòmica.

  El mètode es basa en un politop de N+1 vèrtex i N dimensions (p.e.: per una dimensió un segment, per dues un triangle, i per tres un tetraedre) i en la propietat de que la solució òptima d'un problema de PL es troba al vèrtex o frontera del domini de punts factibles en el politop. L'algoritme avança fent una cerca seqüencial sobre aquests vèrtexs per trobar l'òptim.

- **PLE** o **Programació Lineal Entera**

  Les variables del problema estan al domini dels Enters.

  En un cas general, els vèrtexs de la regió factible no tenen per què ser nombres Enters. En conseqüència, la solució òptima es trobarà a l'interior de la regió factible, per tant, el mètode Simplex, emprat directament sobre el problema, no proporcionarà una solució òptima.

  Tot i que el nombre de possibles solucions d'un model de PLE és finit, *2N* solucions per a un problema amb *N* variables pot ser impossible de manipular (per un problema mitjà, de 30 variables, hi ha 1073741924 solucions possibles).

  És per això que s'ha d'aplicar també altres mètodes com Ramificació i Poda, un algoritme per trobar solucions òptimes basat en recórrer un arbre de solucions on cada branca porta a una solució posterior a l'actual i detectar quines ramificacions tenen solucions que ja no són òptimes per tallar-les.

  Per un problema de PLE es pot (amb complexitat NP-complet):

  1. Aplicar el mètode Simplex al problema de PL associat (relaxant la restricció de que les variables siguin enteres)
  2. Aplicar el mètode de Ramificació per crear un espai de solucions a partir de l'arrodoniment a l'alça i a la baixa de les solucions de Simplex.
  3. Aplicar la Poda basant-nos en un problema de mínims /màxims per a complir la funció objectiu.

- **PLM** o **Programació Lineal Mixta**

  Les variables del problema poden ser nombres Enters o Reals.

  La complexitat depèn del tipus de problema que es vulgui resoldre. L'eina que s'ha desenvolupat treballa amb PLM.

2.3 Programació Lineal i Xarxes de Petri

En aquest punt es vol exposar les motivacions de l'aplicació de la PL en l'estudi de les propietats de les XdP. La informació que es dóna prové, principalment, de l'article "*Basic Linear Algebraic Techniques for Place/Transition Nets*", de Jörg Desel (3).

La **motivació** principal de l'ús de l'àlgebra lineal com a mètode aproximatiu per conèixer el comportament de XdP és la dificultat que pot tenir el seu comportament. La limitació de les tècniques enumeratives respecte de la mida del model fa que l'ús de tècniques estructurals sigui molt més útil en molts casos.

La PL pot donar condicions suficients i/o necessàries per al compliment de propietats en una XdP marcada.

Els algoritmes per solucionar problemes d'àlgebra lineal poden ser més o menys eficients segons el domini en què s'apliquen. En aquests termes, tal i com s'ha vist anteriorment, solucions racionals són molt més fàcils d'obtenir que solucions enteres o naturals.

Totes les tècniques de PL aplicades a les XdP es basen en el fet que el dispar d'una transició sempre provoca el mateix efecte: treu marques del/s lloc/s del pre-conjunt i posa marques al/s lloc/s del post-conjunt, la resta de llocs es mantenen igual.

Els problemes de PL que es resolen amb aquesta eina deriven, en gran part, de l'equació de marcatge, que genera un conjunt d'inequacions, el sistema a resoldre per verificar les propietats. Anomenarem a aquestes restriccions, **restriccions de fila**. A més, es pot afegir restriccions sobre els valors que poden prendre les variables, com per exemple, que siguin Enters; a aquestes restriccions les anomenarem **restriccions de columna**.

Un marcatge no és abastable si no té solució per a l'equació de marcatge. En canvi, poden existir marcatges inabastables que sí tinguin solució entera per l'equació de marcatge i això és degut a la limitació d'aquesta per representar XdP.

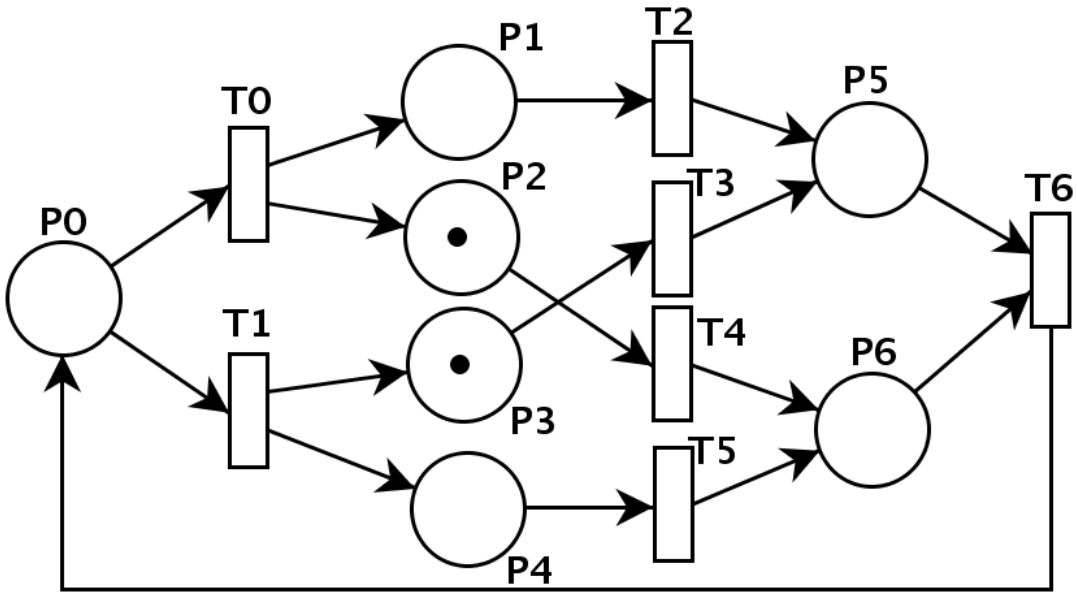


Figura 17: XdP que dóna lloc a resultats espuris per a l'equació de marcatge

De la xarxa de la Figura 17 podem extreure la següent informació:

- Marcatge inicial: $M_0 = (0, 0, 1, 1, 0, 0, 0)$
- Equació de marcatge:

$$[0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0] - [0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0] = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} \cdot x$$

Es vol verificar si el marcatge $M_d = (0, 1, 0, 0, 1, 0, 0)$ és abastable.

Si resolem l'equació de marcatge, obtenim la solució $x = (1, 1, 0, 2, 2, 0, 2)$.

En canvi, si apliquem qualsevol seqüència de dispars possible que satisfaci aquest vector de comptatge de dispars (T_3 , T_4, T_6, $\{T_0, T_4, T_1, T_3\}$, T_6), l'únic marcatge que podem obtenir és $M_d' = (2, 1, 0, 0, 1, 0, 0)$.

L'equació de marcatge no caracteritza el conjunt dels marcatges abastables en una XdP i per tant podrem obtenir resultats espuris per que el nombre de marques en un lloc durant cap moment de l'execució de la xarxa pot ser negatiu, en canvi en la resolució de l'equació de marcatge això no es pot assegurar.

Existeixen, però, altres tècniques per millorar l'equació de marcatge entres les que es troben, tal i com se n'ha parlat dintre de l'apartat de mètodes d'anàlisi (2.1.3 d'aquest document), els invariants. A l' "Annex 1" n'hi ha una ampliació i la introducció a una altra tècnica que són els predicats estables.

2.4 Propietats verificables al sistema

El sistema permet verificar un conjunt de propietats sobre una XdP que es poden relacionar mitjançant operadors. La majoria d'aquestes propietats són les restriccions que es posen a un problema de PL en forma d'inequacions, d'altres es resolen independentment, però també estan basades en equacions de PL.

2.4.1 Propietats

Es pot dividir el tipus de propietats verificables per l'eina segons:

- Restriccions sobre les solucions: generen restriccions sobre esdeveniments que han de passar a la XdP.
- Propietats d'una XdP: donen resposta sobre el compliment de certes propietats a la xarxa.

De vegades ens pot interessar veure el comportament de les xarxes per blocs, que anomenarem traces. Una **traça** és un conjunt d'esdeveniments que s'executa en bloc i que pot englobar propietats que s'apliquen a tot el conjunt. Així, hi haurà moltes de les propietats que hauran d'anar parametritzades a la traça a la qual s'apliquen.

L'eina treballa amb una **propietat per defecte** i és que els valors de les variables a la solució del problema han de ser positius.

A continuació hi ha la definició de cadascuna de les propietats, amb la seva especificació, on s'inclou, segons el cas:

- Codificació: manera d'escriure-la a l'eina d'interfície gràfica.
- Paràmetres: descripció dels paràmetres que té la funció.
- Condició: fet que s'ha de complir per que la propietat es compleixi.
- Equacions PL: conjunt d'equacions que s'afegeixen al problema de PL, només per les restriccions de fila.

S'utilitza la XdP que hi ha a continuació per il·lustrar amb un model cadascuna de les propietats que es poden verificar; el model s'anirà de forma que aquest es vagi incrementant amb cada restricció que s'explica:

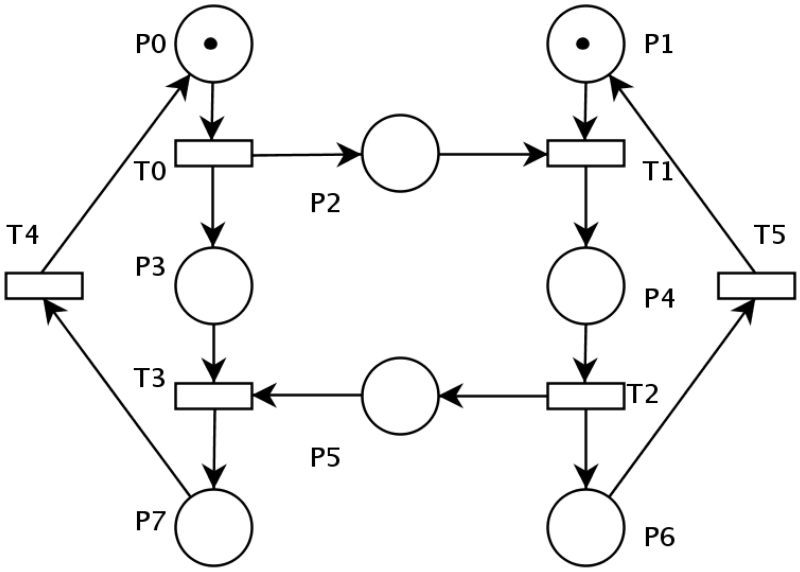


Figura 18: Xarxa de Petri que representa un protocol de comunicació

D'aquest model podem treure la seva matriu incidència i el marcatge inicial:

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix} \qquad M_0 = (1\ 1\ 0\ 0\ 0\ 0\ 0\ 0)$$

També extraiem les equacions bàsiques del model obtingudes a partir de l'equació de marcatge que obliguen a la solució porti a un marcatge on tot els llocs tenen un valor positiu, $\boxed{Ax \geq -M_0}$ (suposem que es crea un model amb una sola traça):

	T0	T1	T2	T3	T4	T5		
R1	-1	0	0	0	1	0	>=	-1
R2	0	-1	0	0	0	1	>=	-1
R3	1	-1	0	0	0	0	>=	0
R4	1	0	0	-1	0	0	>=	0
R5	0	1	-1	0	0	0	>=	0
R6	0	0	1	-1	0	0	>=	0
R7	0	0	1	0	0	-1	>=	0
R8	0	0	0	1	-1	0	>=	0
Tipus	Real	R	R	R	R	R		
Lim sup	∞	∞	∞	∞	∞	∞		
Lim inf	0	0	0	0	0	0		

A partir d'ara a aquest conjunt de restriccions bàsiques les anomenarem **PF** (Propietats per defecte)

Quan aquesta part no variï, se'n farà menció com **Lims** (referent als límits sobre les variables)

Quan s'afegeixin restriccions de fila, al futur s'anomenen $\mathbf{RF_x}$.

Restriccions sobre les solucions

- **Disparar** una transició

Definició	S'afegeix com a restricció a la solució del problema de PL que una transició es dispari al menys una vegada durant una traça.
Codificació	`fire(tr,id_T)`
Paràmetres	- tr: traça en la qual s'ha de disparar la transició. - id_T: identificador de la transició que s'ha de disparar.
Exemple	El model afegint la restricció de disparar la transició T0 queda: T0 T1 T2 T3 T4 T5 PF Tipus R R R R R R Lim sup ∞ ∞ ∞ ∞ ∞ ∞ Lim inf 1 0 0 0 0 0

- **Disparar** una transició **N vegades**

Definició	S'afegeix com a restricció a la solució del problema de PL que una transició es dispari N vegades durant una traça.
Codificació	`nfire(tr,id_T,n_fires)`
Paràmetres	- tr: traça en la qual s'ha de disparar la transició. - id_T: identificador de la transició que s'ha de disparar. - n_fires: nombre de vegades que s'ha de disparar la transició.
Exemple	El model afegint la restricció de disparar la transició T1 dues vegades queda: T0 T1 T2 T3 T4 T5 PF Tipus R R R R R R Lim sup ∞ 2 ∞ ∞ ∞ ∞ Lim inf 1 2 0 0 0 0

- **Silenciar** una transició

Definició	S'afegeix com a restricció a la solució del problema de PL que una transició no es dispari mai durant una traça.
Codificació	`silent(tr,id_T)`
Paràmetres	- tr: traça en la qual no s'ha de disparar la transició. - id_T: identificador de la transició que no s'ha de disparar.
Exemple	El model afegint la restricció de silenciar la transició T4 queda: T0 T1 T2 T3 T4 T5 PF Tipus R R R R R R Lim sup ∞ 2 ∞ ∞ 0 ∞ Lim inf 1 2 0 0 0 0

- **Activar** una transició

Definició	S'afegeix com a restricció a la solució del problema de PL que una transició estigui activa al final de l'execució d'una traça.
Codificació	`enable(tr,id_T)`
Paràmetres	- tr: traça al final de la qual la transició ha d'estar activa. - id_T: identificador de la transició que haurà d'estar activa.
Condició	Cada lloc que es troba al pre-conjunt de la transició ha de tenir, com a mínim una marca.
Equacions PL	$M_0 + Ax \geq M$ on M és un vector que conté 1ns a totes les posicions que representen els llocs que són pre-conjunt de la transició, i 0s a la resta.
Exemple	El model afegint la restricció que la transició T3 estigui activa (RF_1) queda: T0 T1 T2 T3 T4 T5 PF R9 -1 0 0 0 1 0 >= -1 R10 0 -1 0 0 0 1 >= -1 R11 1 -1 0 0 0 0 >= 0 R12 1 0 0 -1 0 0 >= 1 R13 0 1 -1 0 0 0 >= 0 R14 0 0 1 -1 0 0 >= 1 R15 0 0 1 0 0 -1 >= 0 R16 0 0 0 1 -1 0 >= 0 Lims

- **Desactivar** una transició

Definició	S'afegeix com a restricció a la solució del problema de PL que una transició estigui inactiva al final de l'execució d'una traça.
Codificació	`disable(tr,id_T)`
Paràmetres	- tr: traça al final de la qual la transició ha d'estar inactiva. - id_T: identificador de la transició haurà d'estar inactiva.
Condició	De tots els llocs que formen part del pre-conjunt d'una XdP, al menys un ha d'estar buit.
Equacions PL	Aquesta propietat requereix de la creació de tants models com combinacions possibles en què el pre-conjunt de la transició té algun lloc sense marcar. Degut a que la complexitat de la comprovació augmenta significativament, fem més forta aquesta condició i demanem que cap lloc del pre-conjunt de la transició estigui marcat. $M_0 + Ax \leq M$ on M és un vector que conté 0's a totes les posicions que representen els llocs que són al pre-conjunt de la transició, i infinit a la resta.

Exemple	El model afegint la restricció d'activar T3 (RF_2) és: T0 T1 T2 T3 T4 T5 PF RF_1 R17 -1 0 0 0 1 0 <= ∞ R18 0 -1 0 0 0 1 <= ∞ R19 1 -1 0 0 0 0 <= ∞ R20 1 0 0 -1 0 0 <= ∞ R21 0 1 -1 0 0 0 <= ∞ R22 0 0 1 -1 0 0 <= ∞ R23 0 0 1 0 0 -1 <= 0 R24 0 0 0 1 -1 0 <= ∞ Lims

- Un lloc ha de tenir **N marques**

Definició	S'afegeix com a restricció a la solució del problema de PL que el nombre de marques d'un lloc específic al final de l'execució d'una traça ha de ser igual a N.
Codificació	`ntokens(tr,nt,id_P)`
Paràmetres	- tr: traça al final de la qual el lloc ha de tenir el nombre de marques indicat. - nt: nombre de marques que ha de tenir el lloc al final de l'execució de la traça. - id_P: lloc que es vol restringir.
Condició	El marcatge final per una traça conté a la posició del lloc indicat al menys N marques.
Equacions PL	$M_{0i} + Ax_i = M_i$ on M0 és el marcatge inicial del lloc a restringir, Ax_iés el resultat del producte de la matriu d'incidència pel vector de dispars a la fila *i*, i M és nt. El model afegeix $\forall j \in P: j \neq i$ la restricció $M_{0j} + Ax_j = M_j$. Això es fa per facilitar el tractament que l'eina ha de fer sobre el model a l'hora d'afegir i eliminar restriccions.
Exemple	El model afegint la restricció que el nombre de marques del lloc P4 sigui 1 (RF_3) queda: T0 T1 T2 T3 T4 T5 PF RF_1 RF_2 R25 -1 0 0 0 1 0 >= -1 R26 0 -1 0 0 0 1 >= -1 R27 1 -1 0 0 0 0 >= 0 R28 1 0 0 -1 0 0 >= 0 R29 0 1 -1 0 0 0 = 1 R30 0 0 1 -1 0 0 >= 0 R31 0 0 1 0 0 -1 >= 0 R32 0 0 0 1 -1 0 >= 0 Lims

- La solució ha de ser **cíclica**

Definició	S'afegeix com a restricció a la solució del problema de PL que sigui cíclica, és a dir, que torni al marcatge inicial.
Codificació	`cyclicsolution(tr)`
Paràmetres	- tr: traça que ha de ser cíclica.
Condició	La solució obtinguda ha de ser cíclica, és a dir, ha de portar una altra vegada al marcatge inicial.
Equacions PL	Per verificar aquesta propietat cal assegurar que $A \cdot x = 0$.
Exemple	El model afegint la restricció que la solució sigui cíclica (RF_4) queda:

	T0	T1	T2	T3	T4	T5		
				PF				
				RF_1				
				RF_2				
				RF_3				
R26	-1	0	0	0	1	0	=	0
R27	0	-1	0	0	0	1	=	0
R28	1	-1	0	0	0	0	=	0
R29	1	0	0	-1	0	0	=	0
R30	0	1	-1	0	0	0	=	0
R31	0	0	1	-1	0	0	=	0
R32	0	0	1	0	0	-1	=	0
R33	0	0	0	1	-1	0	=	0
				Lims				

- La solució ha de ser **entera**

Definició	S'afegeix com a restricció a la solució del problema de PL que sigui entera.
Codificació	`intsolution(tr)`
Paràmetres	- tr: traça que ha de tenir solució entera.
Exemple	El model afegint la restricció que la solució sigui entera queda:

	T0	T1	T2	T3	T4	T5
			PF			
			RF_1			
			RF_2			
			RF_3			
			RF_4			
Tipus	Enter	E	E	E	E	E
Lim sup	∞	2	∞	∞	0	∞
Lim inf	1	2	0	0	0	0

- La solució ha de ser **binaria**

Definició	S'afegeix com a restricció a la solució del problema de PL que la tots els valors siguin binaris (si existeix una altra restricció sobre el valor de la transició predomina l'altra restricció).
Codificació	`intsolution(tr)`
Paràmetres	- tr: traça que ha de tenir solució entera.
Exemple	El model afegint la restricció que la solució sigui entera queda: T0 T1 T2 T3 T4 T5 PF RF_1 RF_2 RF_3 RF_4 Tipus Enter E E E E E Lim sup 1 2 1 1 0 1 Lim inf 1 2 0 0 0 0 Com es pot veure, els valors del límit superior que fins ara eren infinit queden substituïts pel valor 1. La resta es manté.

Propietats d'una XdP

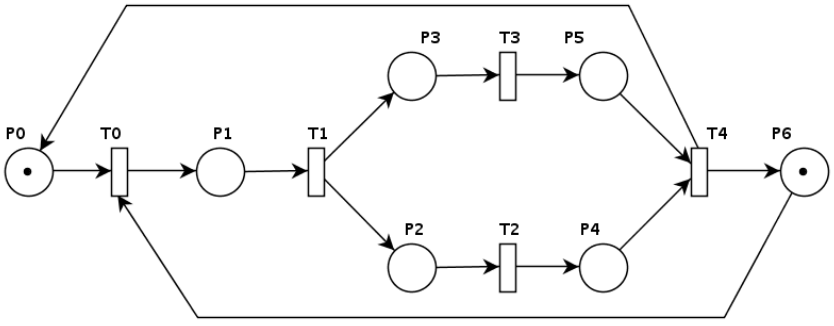
De les següents propietats, l'única que incrementa el model de PL que s'ha mostrat fins al moment és l'abastabilitat; la resta de propietats es comproven de forma independent.

- **Exclusió mútua** entre dos llocs

Definició	Es verifica si hi ha exclusió mútua entre dos llocs de la XdP, és a dir, si no existeix cap marcatge abastable a la XdP on els dos llocs estiguin marcats.
Codificació	mutualexclusion(id_P1,idP2)
Paràmetres	- id_P1: identificador d'uns dels llocs a verificar. - id_P2: identificador de l'altre lloc a verificar.
Fórmula	ntokens(1,1,id_P1)&ntokens(1,1,id_P2) Si existeix solució per aquesta restricció, llavors no hi ha exclusió mútua entre els dos llocs. Altrament, sí que n'hi ha.
Exemple	Comprovem si hi ha exclusió mútua entre els llocs P5 i P7. Es fa de forma independent a la resta del model, creant-ne un de nou amb la mateixa restricció per defecte. La fórmula a verificar és: ntokens(1,1,P5)&ntokens(1,1,P7). El model resultant és: T0 T1 T2 T3 T4 T5 PF R9 -1 0 0 0 1 0 >= -1 R10 0 -1 0 0 0 1 >= -1

R11	1	-1	0	0	0	0	>=	0
R12	1	0	0	-1	0	0	>=	0
R13	0	1	-1	0	0	0	>=	0
R14	0	0	1	-1	0	0	>=	1
R15	0	0	1	0	0	-1	>=	0
R16	0	0	0	1	-1	0	>=	0
R17	-1	0	0	0	1	0	>=	0
R18	0	-1	0	0	0	1	>=	0
R19	1	-1	0	0	0	0	>=	0
R20	1	0	0	-1	0	0	>=	0
R21	0	1	-1	0	0	0	>=	0
R22	0	0	1	-1	0	0	>=	0
R23	0	0	1	0	0	-1	>=	0
R24	0	0	0	1	-1	0	>=	1
Lims								

- **Concurrència** de dues transicions

Definició	Es verifica si dues transicions són concurrents després d'aplicar a l'equació de marcatge un vector de dispars, és a dir, si una vegada disparat el vector indicat, qualsevol de les dues transicions es pot disparar abans, paral·lelament o després que l'altra.
Codificació	concurrence(t_1,...,t_m,id_T1,id_T2)
Paràmetres	- t_1,...,t_m: vector de comptatge de dispars després del qual s'ha de complir la propietat de concurrència - id_T1: identificador d'una de les transicions a verificar. - id_T2: identificador de l'altra transició a verificar.
Fórmula	La transició y i la transició z estan actives i $\|\cdot y\| + \|\cdot z\| \leq \sum_{p \in \cdot yU \cdot z} M[p]$ on M[p] és el marcatge obtingut després de disparar la seqüència indicada.
Exemple	Per il·lustrar la concurrència entre dues transicions introduirem un altre exemple, ja que per l'anterior no es dóna cap cas en què dues transicions estiguin actives al mateix temps, una condició necessària de la concurrència.  Figura 19: Xarxa de Petri Comprovem si hi ha concurrència entre les transicions T2 i T3 després d'executar una seqüència de dispars representada pel vector $x = (1 \quad 1 \quad 0 \quad 0 \quad 0)$. Primer es comprova que les dues transicions estiguin actives: a

	$M_p = Ax + M_0 = (0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)$ els llocs que són pre-conjunt de T2 i de T3 estan marcats. A continuació, es comprova si es compleix la condició de concurrència: $1 + 1 \leq 2$. Es compleix, per tant les dues transicions són concurrents.

- **Conflicte** entre dues transicions

Definició	Es verifica si hi ha conflicte entre dues transicions després d'aplicar a l'equació de marcatge un vector de dispars predefinit, és a dir, si una vegada disparat el vector indicat, es pot disparar, o bé una transició, o bé l'altra, però no les dues.
Codificació	conflict(t_1,...,t_m,id_T1,id_T2)
Paràmetres	- t_1,...,t_m: vector de comptatge de dispars després del qual s'ha de complir la propietat de concurrència entre les dues transicions. - id_T1: identificador d'una de les transicions a verificar. - id_T2: identificador de l'altra transició a verificar.
Fórmula	La transició y i la transició z estan actives i $\|\cdot y\| + \|\cdot z\| > \sum_{p \in \cdot yU \cdot z} M[p]$ on M[p] és el marcatge obtingut després de disparar la seqüència indicada.
Exemple	Seguin amb l'exemple introduït a la propietat de concurrència, comprovem ara si hi ha conflicte entre les transicions T0 i T3 després d'executar una seqüència de dispars representada pel vector $x = (1 \quad 0 \quad 0 \quad 0 \quad 0)$. Primer es comprova que les dues transicions estiguin actives: a $M_p = Ax + M_0 = (0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)$ el lloc que és pre-conjunt de la transició T3, per tant, entre les dues transicions no hi ha conflicte.

- **Abastabilitat** d'un marcatge

Definició	Es verifica si un marcatge és abastable per a una traça concreta.
Codificació	reachable(tr,p_1,...,p_n)
Paràmetres	- tr: traça al final de la qual el marcatge ha de ser abastable. - p_1,...,p_n: vector que conté a cada posició el nombre de marques que té el lloc que representa.
Condició	El marcatge final ha de ser l'indicat.
Equacions PL	$M_0 + Ax = M$ on M és el vector d'entrada.
Exemple	Seguint amb el model del principi, si ara s'afegeix la restricció que el marcatge final sigui $M = (0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)$ (RF_5) queda:

	T0	T1	T2	T3	T4	T5		
				PF				
				RF_1				
				RF_2				
				RF_3				
				RF_4				
R34	-1	0	0	0	1	0	=	-1
R35	0	-1	0	0	0	1	=	-1
R36	1	-1	0	0	0	0	=	0
R37	1	0	0	-1	0	0	=	0
R38	0	1	-1	0	0	0	=	1
R39	0	0	1	-1	0	0	=	0
R40	0	0	1	0	0	-1	=	0
R41	0	0	0	1	-1	0	=	0
				Lims				

- Un vector és un **S-invariant**

Definició	Es verifica si un vector és una S-invariant d'una XdP, és a dir, si es compleix que el conjunt de llocs que conté el vector manté el nombre total de marques després de disparar qualsevol combinació factible de transicions.
Codificació	sinvariant(p_1,...,p_n)
Paràmetres	- p_1,...,p_n: vector que conté a cada posició el nombre de marques que té el lloc que representa.
Fórmula	$x \cdot A = 0$, on x és el invariant a verificar.
Exemple	Continuant amb l'exemple introduït a la propietat de concurrència, comprovem si el vector $x = (1 \quad 1 \quad 1 \quad 1 \quad 1)$. Es calcula: $xA = (1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0) \cdot \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix} =$ $(0 \quad 0 \quad -1 \quad 0 \quad 1) \neq (0 \quad 0 \quad 0 \quad 0 \quad 0)$. El resultat indica que el vector no és un S-invariant de la xarxa.

- Un vector és un **T-invariant**

Definició	Es verifica si un vector és una T-invariant d'una XdP, és a dir, si conté una seqüència de dispars que dóna com a marcatge resultant el marcatge inicial.
Codificació	tinvariant(t_1,...,t_m)
Paràmetres	- t_1,...,t_m: vector que conté a cada posició el nombre de dispars de la transició que representa.

Fórmula	$A \cdot y = 0$, on x és el vector a verificar.
Exemple	Continuant amb l'exemple introduït a la propietat de concurrència, comprovem si el vector $y = (1 \quad 1 \quad 1 \quad 1 \quad 1)$. Es calcula: $Ay = \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot (1 \quad 1 \quad 1 \quad 1 \quad 1) =$ $(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$ El resultat indica que el vector és un T-invariant de la xarxa.

2.4.2 Traça

Tal i com s'ha introduït anteriorment, una **traça** és un conjunt d'esdeveniments a la XdP que s'executen en bloc i que poden tenir propietats comunes.

En el cas que es vulgui comprovar les propietats per diferents traces, el sistema crea un model on el nombre de variables és igual al nombre de transicions del sistema multiplicat pel nombre de traces.

La matriu $\mathcal{A}$ que s'aplica a l'equació de marcatge és una matriu de tantes files com nombre de llocs tingui la xarxa, i tantes columnes com nombre de transicions multiplicat pel nombre de traces. Podem representar aquesta matriu com la concatenació de tantes matrius de la mida de la matriu d'incidència com traces es vulgui resoldre, i el seu contingut dependrà de la traça a la que s'hagi d'aplicar la restricció. Per exemple, suposem una xarxa amb la matriu d'incidència A, on volem verificar propietats sobre 3 traces:

- Per la primera traça, la matriu $\mathcal{A}$ serà:

  $\mathcal{A} = (A \mid 0 \mid 0)$, on 0 és una matriu de la mida de A però amb tots els valors a 0.

- Per la segona traça, la matriu $\mathcal{A}$ serà:

  $\mathcal{A} = (A \mid A \mid 0)$

- Per la tercera traça, la matriu $\mathcal{A}$ serà:

  $\mathcal{A} = (A \mid A \mid A)$

Per posar un exemple simple, si a la XdP dels exemples anteriors volem verificar si és possible que a la primera traça la transició T5 no es dispari i la transició T0 sí que ho faci, i que a la segona traça la transició T0 es torni a disparar:

- S'executarà la comanda: *silent(1,T5)&fire(1,T0)&fire(2,T0)*

- I s'obtindrà un model com el següent:

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11		
R1	-1	0	0	0	1	0	0	0	0	0	0	0	>=	-1
R2	0	-1	0	0	0	1	0	0	0	0	0	0	>=	-1
R3	1	-1	0	0	0	0	0	0	0	0	0	0	>=	0
R4	1	0	0	-1	0	0	0	0	0	0	0	0	>=	0
R5	0	1	-1	0	0	0	0	0	0	0	0	0	>=	0
R6	0	0	1	-1	0	0	0	0	0	0	0	0	>=	0
R7	0	0	1	0	0	-1	0	0	0	0	0	0	>=	0
R8	0	0	0	1	-1	0	0	0	0	0	0	0	>=	0
R9	-1	0	0	0	1	0	-1	0	0	0	1	0	>=	-1
R10	0	-1	0	0	0	1	0	-1	0	0	0	1	>=	-1
R11	1	-1	0	0	0	0	1	-1	0	0	0	0	>=	0
R12	1	0	0	-1	0	0	1	0	0	-1	0	0	>=	0
R13	0	1	-1	0	0	0	0	1	-1	0	0	0	>=	0
R14	0	0	1	-1	0	0	0	0	1	-1	0	0	>=	0
R15	0	0	1	0	0	-1	0	0	1	0	0	-1	>=	0
R16	0	0	0	1	-1	0	0	0	0	1	-1	0	>=	0
Tipus	R	R	R	R	R	R	R	R	R	R	R	R		
Lim sup	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞		
Lim inf	1	0	0	0	0	0	1	0	0	0	0	0		

La xarxa tenia 6 transicions i com estem treballant amb dues traces, tenim 12 variables de *T0* a *T11*. Hi ha 16 restriccions de fila, de *R1* a *R16*, que corresponen a la restricció per defecte aplicada a les dues traces: 8 llocs per 2 traces. Finalment, a la part inferior hi ha les restriccions de columna, on es pot veure la representació de les tres propietats indicades anteriorment.

2.4.3 Operadors

Tot el conjunt de propietats prèviament explicat es pot verificar de forma conjunta a partir de relacions amb tres operadors: **negació**, **conjunció** i **implicació**.

La negació és l'operador amb més **prioritat**, així que serà el que s'executarà primer. La conjunció i la implicació tenen menys prioritat que la negació, però la mateixa entre elles, així, si no s'especifica d'una altra manera, la prioritat vindrà donada per l'ordre d'esquerra a dreta de les propietats. L'aplicació permet modificar aquest ordre de prioritats amb l'ús de claudàtors.

Tractament dels operadors:

- La negació es tracta abans d'introduir la restricció al model.
- La concatenació de propietats és trivial en el sentit que només cal afegir la restricció corresponent al model o solucionar la propietat (si és una individual).
- La teoria per la resolució de la operació d'implicació s'ha extre de l'article "*Basic Linear Algebraic Techniques for Place/Transition Nets*" de Jörg Desel (3). El seu tractament és el més prioritari i elaborat.

Sigui φ un predicat lineal del tipus $u \cdot m \geq v_i$. El conjunt $\{\varphi_1, \dots, \varphi_n\}$ implica φ si i només si el següent sistema d'inequacions no té solució sobre els Naturals:

$$u_1 \cdot m \leq v_1,$$
$$u_2 \cdot m \leq v_2,$$
$$\dots$$
$$u_n \cdot m \leq v_n,$$
$$u \cdot m \geq v + 1.$$

Així doncs, per resoldre les propietats d'implicació cal comprovar que no existeix cap solució al sistema que accepti un model que té com a restriccions totes les de la part esquerra de la implicació i la negació de la part dreta.

Si es verifica que la implicació es compleix podem transformar l'operador en una conjunció, ja que si part esquerra s'ha de complir i quan es compleix, s'ha de complir la part dreta, llavors s'ha de complir tot el conjunt.

El model es va llegint d' esquerra a dreta i es va solucionant segons aquest ordre i l'ordre de prioritats establert. En el moment que una propietat o conjunt de propietats no té solució, les següents no es validen ja que segur que el conjunt no té solució.

Cal tenir en compte que l'operació de disjunció no està implementada i això fa que certes estructures que es podrien construir segons el que s'ha explicat no siguin verificables:

- Negació d'algunes propietats

  Hi ha propietats que generen més d'un model quan es neguen i en aquesta eina es treballa amb un model únic. També, hi ha propietats per les que la seva negació és molt complexa.

  Les operacions que no es poden negar són:

  - N marques: per negar aquesta restricció caldria crear dos models, un que contingui els marcatges en que el lloc a restringir té menys de N marques, i un altre en que els marcatges pel lloc en qüestió tenen més de N marques.
  - N dispars: igual que abans, per negar aquesta restricció cal crear dos models, un pel que el nombre de dispars de la transició sigui major a N, i un altre pel que sigui menor.
  - Ciclicitat: no es suficient amb verificar que $Ax \neq 0$, ja que aquesta propietat l'únic que assegura és que no es torna al marcatge inicial, però pot ser que la xarxa tingui cicles.

  En aquesta part cal tenir en compte que, per la seva definició, la propietat "activar una transició" no s'hauria de poder negar ja que també genera molts models (tots aquells en què al menys un dels llocs del pre-conjunt de la transició en qüestió no està marcat). Tot i això, i coherentment amb la definició de la propietat "desactivar una transició", es permet la seva negació tenint en compte que el que s'obté és el resultat d'una restricció més forta: que cap dels llocs del pre-conjunt de la transició estigui marcat.

  D'altra banda, les propietats de concurrència, conflicte, abastabilitat i invariants no es poden negar per que no té sentit, ja que, si el resultat de l'execució de la propietat és negatiu, llavors la propietat no es compleix.

- Negació d'un conjunt de propietats

La negació d'una conjunció de propietats dóna com a resultat una disjunció de la negació de les propietats: ¬(A ∧ B) = ¬A ∨ ¬B.
Com ja em dit, l'eina no treballa amb el operador de disjunció, és per això que no es permet fer aquest tipus de negacions.

- Implicació on la part dreta té més d'una restricció

  Per exemple, la restricció "*[[fire(1,T3)&fire(1,T5)]=>fire(1,T0)]]*" sobre la XdP que s'exposa a l'apartat anterior, genera el següent model:

	T0	T1	T2	T3	T4	T5		
R1	-1	0	0	0	1	0	>=	-1
R2	0	-1	0	0	0	1	>=	-1
R3	1	-1	0	0	0	0	>=	0
R4	1	0	0	-1	0	0	>=	0
R5	0	1	-1	0	0	0	>=	0
R6	0	0	1	-1	0	0	>=	0
R7	0	0	1	0	0	-1	>=	0
R8	0	0	0	1	-1	0	>=	0
Tipus	R	R	R	R	R	R		
Lim sup	0	∞	∞	∞	∞	∞		
Lim inf	0	0	0	1	0	1		

  No es permet resoldre implicacions on a la part dreta hi hagi més d'una restricció, del tipus A => (B ∧ C) per que això es convertiria en A ∧ (¬B ∨ ¬C) i, com ja em dit, no està disponible l'operador de disjunció.

Seguint amb l'exemple anterior, cal notar la diferència entre les dues fórmules:

- "*[nfire(1,T1,1)=>fire(1,T0)]&[nfire(2,T3,1)&silent(1,T2)=>fire(2,T2)]*"
- "*[nfire(1,T1,1)=>fire(1,T0)]&[[nfire(2,T3,1)&silent(1,T2)]=>fire(2,T2)]*"

La primera no es compleix, ja que, el que s'està comprovant és que:

- no hi ha cap solució a la traça 1 on T1 es dispari 1a vegada i T0 no es dispari i
- la transició T3 es dispara una vegada a la traça dos i
- no hi ha cap solució on T2 no es dispari a la primera traça i sí que es dispari a la segona.

En canvi, per la segona propietat, es modifiquen les prioritats i ara sí que dóna solució, per que el que es comprova és que:

- no hi ha cap solució a la traça 1 on T1 es dispari 1a vegada i T0 no es dispari i
- no hi ha cap solució on es dispari T3 a la segona traça, no es dispari T2 a la primera traça i tampoc a la segona.

3 Especificació

En aquest apartat es defineix quins són els requisits que haurà de satisfer l'aplicació desenvolupada, tant pel que respecta a la llibreria com a l'executable amb interfície gràfica. Es fa una especificació semi-formal dels requisits de l'aplicació i una especificació formal de les funcionalitats i dels models de dades i de comportament.

3.1 Anàlisi de requisits

En l'àmbit de l'Enginyeria del Software, l'anàlisi de requisits engloba aquelles tasques que determinen les necessitats i/o condicions que l'usuari desitja que el sistema compleixi amb la finalitat de solucionar un problema o aconseguir un objectiu.

Existeixen diferents tipus de requisits, els més comuns dels quals són:

- **Requisits funcionals**: descriuen què fa el sistema, quines són les entrades i sortides del mateix i les relacions entre les dues i, fins i tot, les dades i els processos.
- **Requisits no funcionals**: descriuen les qualitats generals que ha de complir el sistema per a poder realitzar les seves funcions.

3.1.1 Requisits funcionals

A continuació s'inclou la llista d'activitats que representa què ha de fer el sistema agrupades per mòduls i fitxers d'entrada i sortida:

- **Entrades**:
  - *XdP*: el sistema requereix un tipus de fitxer concret que ve especificat per l'estàndard PNML[1] i per l'estructura PNTD[2] que s'ha especificat pel reconeixement de les XdP que s'accepta.

[1] **Petri Net Markup Language**

És un concepte basat en el format XML que defineix la total estructura d'un fitxer que conté una XdP.

Degut a l'extensa tipologia de xarxes, es defineix de forma genèrica les característiques d'una XdP mitjançant un document anomenat PNTD[2].

[2] **Petri Net Type Definition**

Document que descriu el format específic amb que es vol modelar un conjunt de XdP.

El sistema ha de respondre correctament a una entrada incorrecta indicant l'error esdevingut i no propagant-lo.
    - *Propietats*: les propietats que es permet verificar han de tenir un format específic que es defineix en una gramàtica i que és validat per la mateixa.
      El sistema ha de contemplar el cas que les propietats no estiguin ben definides i indicar a l'usuari on està l'error sense propagar-lo.
- **Sortides**: el format dels fitxers de registre que genera l'aplicació amb interfície gràfica ha de ser sempre el mateix i representar els errors, els avisos i els missatges informatius d'una forma tipificada i uniforme.
- **XdP**: el mòdul de les XdP ha de complir els requisits,
    - Validar sintàcticament un fitxer amb format XML amb la definició d'una XdP a partir d'un esquema descrit en un altre fitxer amb format XML amb la definició de l'estructura que ha de tenir la descripció de la XdP. També reportar els resultats obtinguts de la validació.
    - Crear una estructura de dades a partir del fitxer XML amb la XdP que contingui les dades necessàries per a la verificació de propietats.
    - Consultar la informació continguda en l'estructura de dades amb la XdP: llocs, transicions, arcs i marques de tota la xarxa.
- **Propietats**: el mòdul de les Propietats està dividit en dos submòduls, un que realitza l'anàlisi sintàctica de les propietats i un altre que conté una estructura de dades amb les propietats que l'usuari vol verificar. Separem els requisits segons aquesta divisió de responsabilitats,
    - Anàlisi de les propietats:
        - Analitzar sintàcticament les propietats que l'usuari ha introduït i reportar els resultats obtinguts.
        - Crear una estructura de dades que guardi les propietats que es vol verificar: quina propietat és, quins paràmetres té, etc.
    - Estructura de dades amb les propietats:
        - Afegir i eliminar propietats.
        - Consultar les propietats i tots els seus paràmetres.
- **PL**: el mòdul de PL ha de complir els requisits,
    - Definir un model d'àlgebra lineal que, com a base, incorpori les restriccions derivades de l'estructura de la XdP.
    - Afegir i eliminar del model les propietats que s'ha establert.
    - Consultar les dades que conté el model.
- **Presentació**: l'eina ha de permetre a l'usuari interactuar amb ella mitjançant dues vies,
    - Un llibreria estàtica que permeti utilitzar els mòduls individualment.
    - Una interfície gràfica que permeti verificar propietats a partir d'un fitxer amb la XdP i una cadena de caràcters amb les propietats.

3.1.2 Requisits no funcionals

Donat que es vol realitzar un programari de qualitat, s'ha especificat un sèrie de requisits no funcionals que permetin complir el major nombre de factors de qualitat del software possible. Aquests requisits es mostren a continuació.

Modularitat

Es defineix la modularitat com la capacitat que té un sistema per ser vist com la unió de diferents parts que interactuen entre sí i que treballen per assolir un objectiu comú.

El sistema ha d'estar dividit en mòduls que agrupin les diferents funcionalitats de l'eina de verificació de propietats. Aquests han de ser independents entre sí (treballar com *caixes negres*) i comunicar-se a través d'unes entrades i unes sortides ben definides.

Aquest factor ens garanteix l'**extensibilitat** del codi, és a dir, que el producte serà fàcil d'adaptar a canvis futurs, ja que només es veurà afectat el mòdul que es modifiqui.

També es té en compte la **reusabilitat** del sistema i, per això, a més d'una aplicació amb interfície gràfica s'ha desenvolupat una llibreria estàtica que pot ser importada des d'un altre codi i usada total o parcialment segons les necessitats del desenvolupador.

Fiabilitat

S'entén com fiabilitat el grau en que s'espera que un programa realitzarà la seva funció amb una certa precisió. En aquests termes podem dir que:

- Es garanteix la **correctesa** del sistema fent proves de cada funcionalitat mitjançant un procés de *procés de integració continua*[3].
- Es manté la **robustesa** ja que es controla els esdeveniments inesperats amb una col·lecció d'errors que no es propaguen i es reporten a l'usuari final.
  En el cas de la llibreria, els mètodes retornen els codis d'error especificats en cada cas per tal que el desenvolupador no propagui els errors i, en el cas de l'executable, la interfície retorna els missatges necessaris a l'usuari per a poder corregir les entrades i deixa tots els registres d'execució en un fitxer de registre.

Eficiència

El sistema ha d'usar la mínima quantitat de recursos i de codi possibles per a realitzar totes les seves funcions. De totes les funcionalitats desenvolupades, la que requereix més recursos és l'ús d'un solucionador de PL i, respecte d'aquest punt, es garanteix que el nombre de crides que es fa a aquesta llibreria és mínim mitjançant uns processos previs de simplificació de l'entrada.

Usabilitat

El disseny ha d'estar centrat en l'usuari, pensant en una eina pràctica, fàcil d'aprendre i d'utilitzar, complint les condicions específiques d'ús. S'ha de tenir en compte, és clar, que l'aplicació requereix una base de coneixement mínima sobre les XdP, però no sobre la resta d'àmbits d'aplicació com poden ser la PL o les eines de reconeixement del llenguatge usades per l'exploració de les propietats que l'usuari vol validar.

Dins d'aquests paràmetres, s'ha construït un sistema en diferents mòduls segons les diferents funcionalitats i s'ha desenvolupat el codi seguint els estàndards del llenguatge, facilitant al màxim l'ús i enteniment de la llibreria; d'altra banda, l'aplicació ha de tenir una

[3] Un **procés d'integració continua** es basa en un procés que:

- S'activa quan hi ha qualsevol canvi al codi font del sistema.
- Recompila tots el processos que utilitzen el codi font que s'ha modificat.
- Si algun falla, es dóna l'error al desenvolupador per tal de poder corregir-lo.

interfície d'usuari clara i fàcil d'usar. També els fitxers d'entrada han de ser fàcils de definir i els de sortida fàcils d'interpretar.

Mantenibilitat

La facilitat de comprendre i adaptar o corregir el software també s'ha de tenir en compte. Per això la documentació i modularitat són aspectes importants.

La documentació del sistema està formada per aquest document que engloba la part tècnica i la part pràctica del projecte. L'usuari podrà trobar el ell un manual d'ús, i el tècnic podrà trobar la informació necessària per entendre el funcionament de l'aplicació i la seva arquitectura, i tindrà eines més que suficients per poder fer modificacions.

Un aspecte important que s'ha tingut en compte durant el desenvolupament és que aquest sistema pugui ser utilitzat per persones que provinguin d'àmbits diferents i tinguin un nivell de coneixement diferent sobre els temes que es tracten. Per això és tant important el desenvolupament del sistema com la documentació del mateix.

3.2 Anàlisi funcional

En aquest punt s'introdueix el model conceptual, els actors, els casos d'ús i la interfície per tal de donar una visió amb un alt nivell d'abstracció de l'estructura del sistema i del comportament de l'aplicació.

A partir de les reunions amb el tutor, els objectius fixats pel projecte i els recursos materials i temporals dels que s'ha disposat, es presenta el següent anàlisi que serveix de referència durant la implementació del projecte i a posteriori per a determinar el nivell d'èxit del mateix.

3.2.1 Model de dades conceptual

El model conceptual és una descripció, potser la més significativa, del domini del problema al qual identifica:

- **Classes d'objectes**: una classe representa un concepte. Descriu un conjunt d'objectes amb les mateixes propietats, comportament i semàntica, i les mateixes relacions amb altres objectes (classes).
  Es simbolitza les classes amb caixes que contenen al seu interior:
  - **Atributs**: són les propietats que comparteixen tots els objectes d'una classe. En aquest diagrama s'indica els més significatius pel disseny i s'expressen com paraules clau amb el tipus del valor que contenen.
  - **Operacions**: són els mètodes que comparteixen tots els objectes d'una classe. També s'indica els més significatius (s'obvia sempre els mètodes constructors, els consultors, etc.) i s'expressen com paraules clau amb els seus paràmetres.
- **Associacions de classes**: una associació entre dues classes representa una relació entre els conceptes que representen i afegeix un rol a cada concepte de l'associació; aquest, a més, pot anar acompanyat d'una cardinalitat.

S'expressa una associació amb una línia que uneix dues classes; aquesta pot tenir un nom, indicat al centre de la línia. Els rols es representen amb noms i les cardinalitats amb números o constants a cada costat de l'associació.

- **Restriccions textuals**: encara que es faci un diagrama molt acurat, no és possible descriure totes les propietats i/o comportaments dels objectes que hi apareixen. En aquests casos s'afegeix en format de text pla totes aquestes característiques.
  Les restriccions es mostren com a frases dins d'etiquetes.
  En el model de dades no s'inclou cap restricció textual. Aquestes es definiran més endavant, al diagrama de classes del disseny, ara no són significatives.

A continuació es mostra l'esquema general de la relació entre els diferents mòduls:

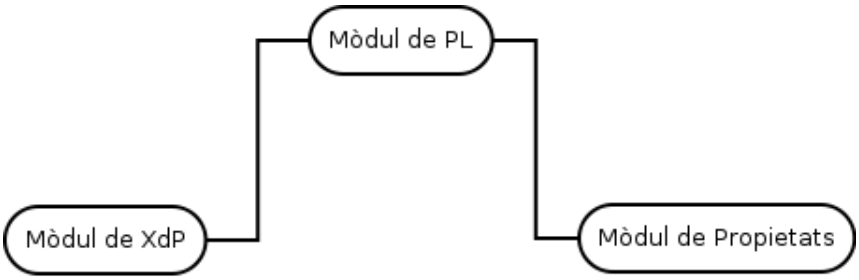


Figura 20: Model de dades general

Als següents punts es mostra l'estructura interna de cada mòdul. Cal tenir en compte no es mostrarà els casos d'ús bàsics, que entenem per funcions creadores i destructores, funcions consultores i "getters" i "setters".

3.2.1.1 Mòdul de XdP

Aquest és el mòdul que s'encarrega d'emmagatzemar les dades sobre la XdP. Queda separat en dues funcionalitats diferents:

1. **Anàlisi sintàctica** de l'arxiu que conté la **XdP** en base a un esquema que defineix quina l'estructura que ha de tenir una xarxa.

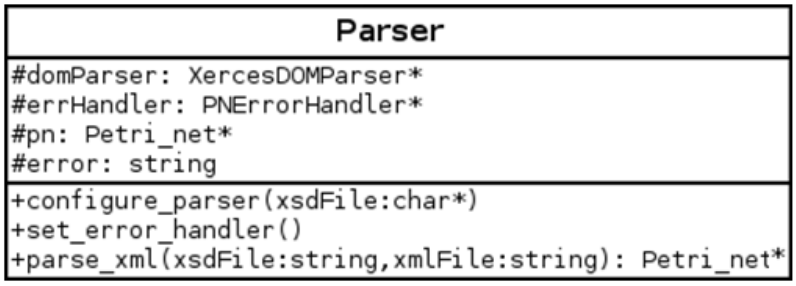


Figura 21: Analitzador sintàctic de la XdP

L' objecte **Parser** representa un analitzador sintàctic i emmagatzema, dinàmicament, informació sobre els nodes que conformen l'XML amb la XdP i sobre els possibles errors que pot contenir el fitxer.

2. **Estructura de dades** per guardar la XdP.

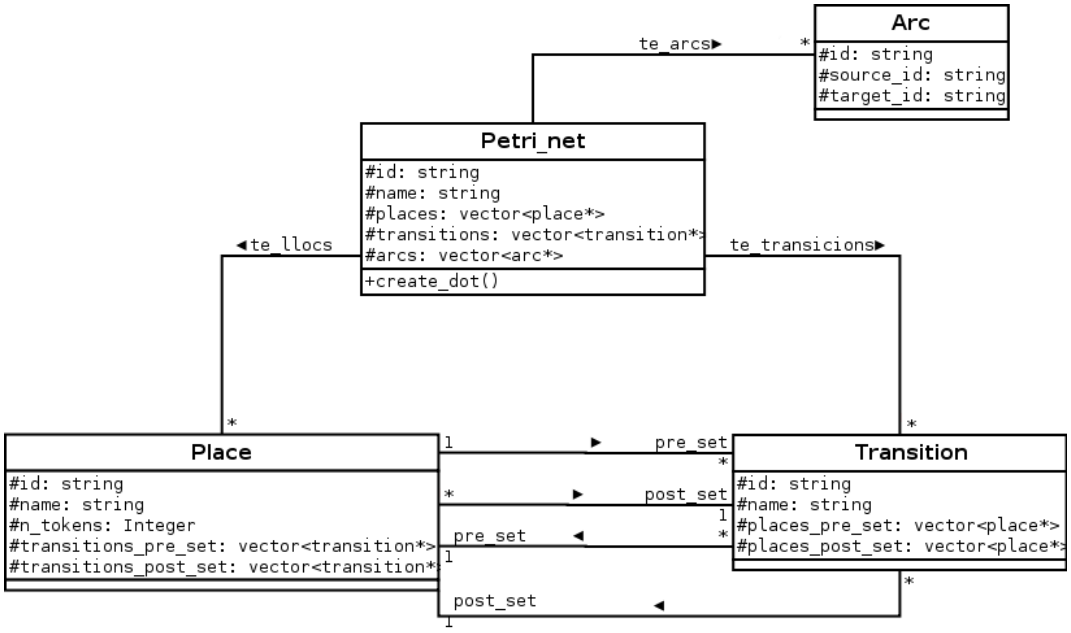


Figura 22: Mòdul de la XdP

A continuació s'explica la raó de ser de cada objecte:

- **Place**
  Aquest objecte representa un lloc i emmagatzema informació relativa a ell com: el seu identificador, el nom i el nombre de marques que conté, així com punters a les transicions que formen part del seu pre-conjunt i a les que formen part del seu post-conjunt.
- **Transition**
  Aquest objecte representa una transició i emmagatzema dades sobre ella com: el seu identificador, el nom i punters als llocs que formen part dels seus pre-conjunt i post-conjunt.
- **Arc**
  Aquest objecte representa les relacions entre llocs i transicions i emmagatzema la informació necessària per descriure-les com: un identificador, els identificadors dels objectes que són inici i fi de l'arc, i els tipus d'aquests objectes (el tipus pot ser lloc o transició).
- **Petri_net**
  Aquest és l'objecte que emmagatzema la informació general de la XdP, com el seu nom i un identificador, i conté punters als diferents objectes que s'ha descrit anteriorment de forma que l'estructura de la xarxa queda completament definida.

3.2.1.2 Mòdul de PL

Aquest és el mòdul que s'encarrega de la comunicació entre la XdP i el solucionador de problemes basat en programació lineal.

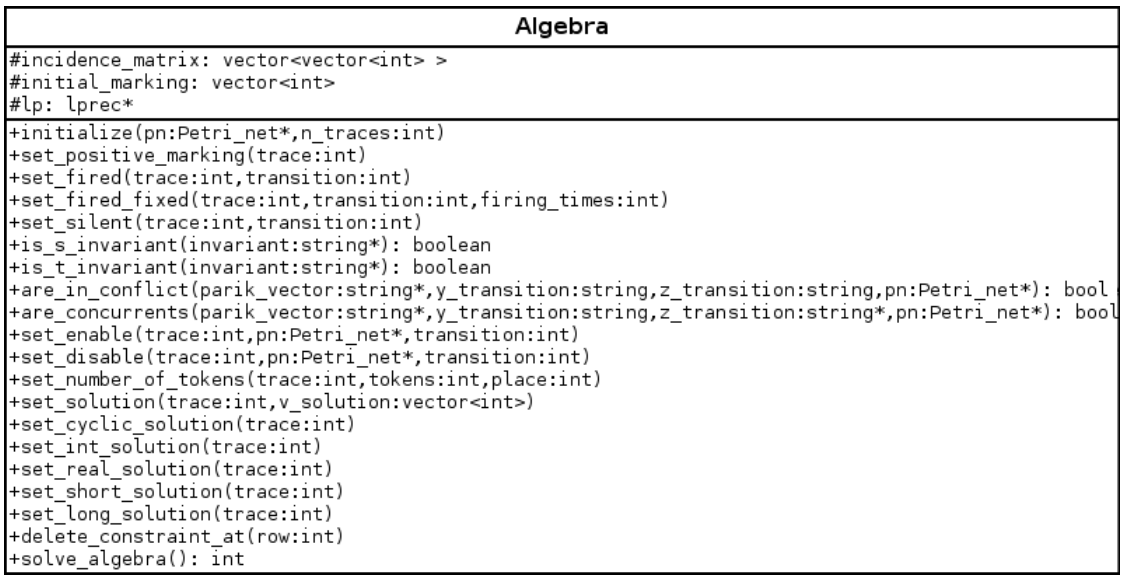


Figura 23: Mòdul de PL

L'objecte **Algebra** conté la informació que l'equació de marcatge extreu en aplicar-se a una XdP i conté un objecte del tipus *lprec* que representa el solucionador de problemes de PL. Aquesta classe processa totes les propietats que el sistema permet verificar sobre una XdP i és qui s'encarrega de fer la verificació.

3.2.1.3 Mòdul de Propietats

El mòdul de Propietats s'encarrega de llegir una entrada amb les propietats a verificar, analitzar la seva correctesa lèxica, sintàctica i semàntica i fer les crides corresponents al mòdul de PL per resoldre el problema de plantejat d'una forma eficient.

Queda separat en dos submòduls diferents:

1. **Estructura de dades** que conté les propietats a verificar

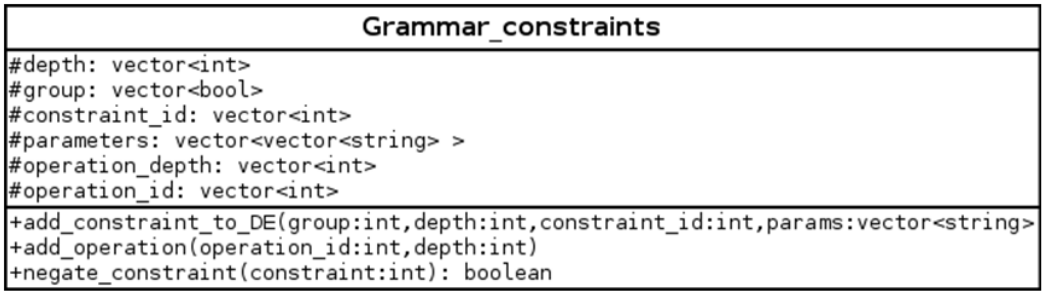


Figura 24: Estructura de Dades de les Propietats

L'objecte **Grammar_constraints** conté les restriccions a verificar; és un lloc d'emmagatzematge intermedi per tal de poder organitzar les restriccions en el mínim nombre de grups possibles amb el màxim nombre de restriccions cadascun i, d'aquesta manera, fer el mínim nombre de crides possible al solucionador de PL. Es guarda

informació de les restriccions (l'identificador, la prioritat i els paràmetres) i de les operacions que relacionen aquestes restriccions (l'identificador i la prioritat).

2. **Anàlisi lèxica, sintàctica i semàntica** de la cadena que conté les propietats que es vol verificar sobre una XdP.

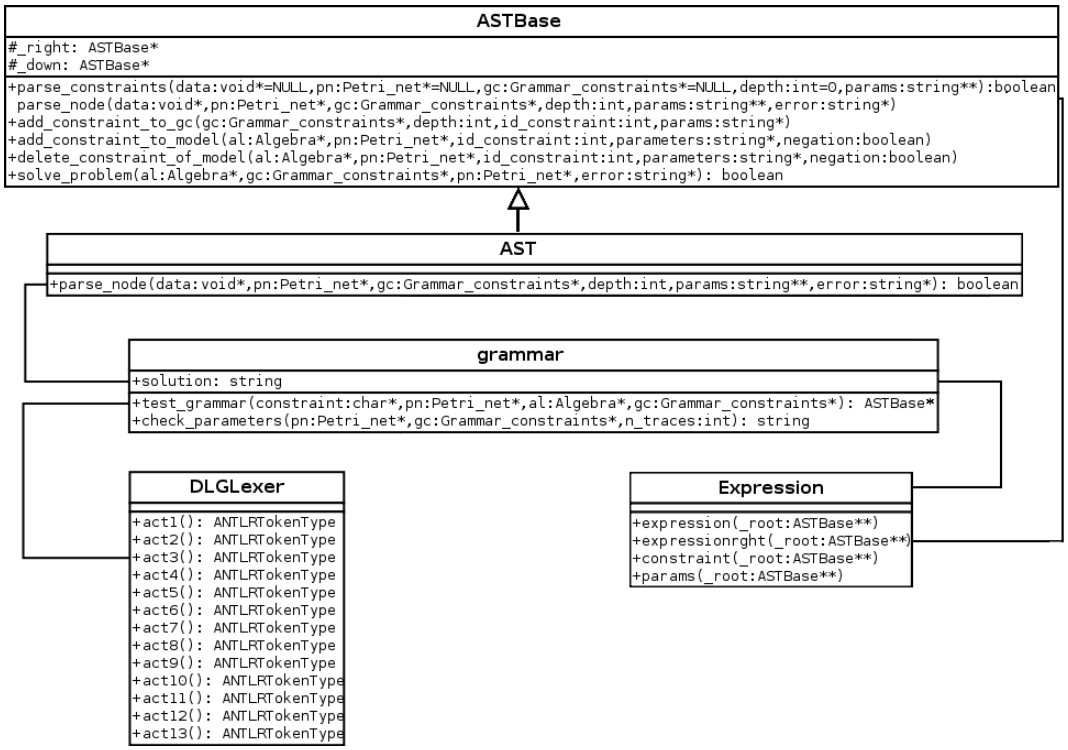


Figura 25: Mòdul per l'anàlisi i solució de Propietats

Els objectes d'aquesta classe són els següents:

- **Expression**
  Objecte que conté l'analitzador sintàctic que s'utilitza per verificar la correctesa de les propietats a verificar. Conté un mètode per cada producció de la gramàtica.
- **DLGLexer**
  Objecte que conté la definició del lèxic que l'analitzador sintàctic accepta com a vàlid. Conté un mètode per a cada token que la gramàtica ha d'acceptar.
- **ASTBase**
  Objecte que conté l'arbre generat per la gramàtica a partir d'una cadena propietats. Aquesta classe s'encarrega de crear l'arbre i, d'una banda crear l'estructura de dades amb les restriccions, i d'altra comunicar-se amb el mòdul de PL per crear el model i solucionar-lo si s'escau.
- **AST**
  Objecte que implementa la lectura dels tokens de l'arbre de propietats.
- **Grammar**
  Objecte que es relaciona amb l'analitzador lèxic i sintàctic i que s'encarrega de l'anàlisi semàntica de la cadena amb les propietats. És l'objecte que, una vegada

comprovada la validesa de les propietats, es comunica amb l'objecte ASTBase per tal de solucionar el problema.

3.2.2 Model de casos d'ús

3.2.2.1 Actors

L'eina està pensada per ser utilitzada per dos tipus d'actor

- **Analitzador de XdP**: és la persona que principalment utilitzarà la interfície d'usuari per a validar propietats sobre XdP.
- **Programador**: és la persona que està capacitada per utilitzar la llibreria estàtica que s'ha desenvolupat i per fer-ne ampliacions.

Tots els actors han de tenir coneixements bàsics sobre les XdP i el seu comportament.

Analitzador

Aquest serà l'usuari que explotarà l'eina a partir de la interfície gràfica.

No ha de tenir coneixements sobre la implementació, únicament sobre les funcionalitats de l'eina i com l'ha d'usar. Amb la descripció de les propietats feta a l'apartat de teoria, l'especificació de la interfície gràfica i el format xml acceptat, els mapes navegacionals del disseny i els exemples d'aquest document té totes les eines necessàries per treballar amb el sistema.

Aquest actor serà capaç de:

- Crear fitxers d'entrada amb el format correcte
- Verificar propietats sobre una XdP,
- Llegir i entendre els fitxers de registre amb els missatges que deixa l'eina durant tot el procés.

Analitzador/programador

Aquest usuari podrà explotar l'eina des de la interfície gràfica i des de la llibreria.

Ha de tenir coneixements de:

- Programació en C++, el llenguatge en què s'ha desenvolupat la llibreria
- Els àmbits de l'eina amb què vol treballar: XdP i/o PL.

Aquest actor serà capaç de:

- Verificar propietats en XdP a partir de la interfície d'usuari o bé usant la llibreria estàtica.
- Utilitzar qualsevol dels mòduls de la llibreria independentment de la resta:
    - Analitzar sintàcticament XdP i cadenes de Propietats.
    - Crear/modificar l'estructura amb la XdP manualment.
    - Crear/modificar l'estructura amb les Propietats manualment.
    - Crear/modificar el model de PL manualment.
- Fer modificacions en el codi de qualsevol dels mòduls

3.2.2.2 Diagrames de casos d'ús

Les possibilitats d'explotació del sistema varien segons si s'utilitza l'eina amb interfície gràfica o la llibreria estàtica.

A continuació es presenta les funcions principals de cadascuna de les eines. Cada cas d'ús representa una seqüència d'esdeveniments que realitza un actor al sistema per obtenir un resultat concret.

3.2.2.2.1 Eina amb interfície gràfica

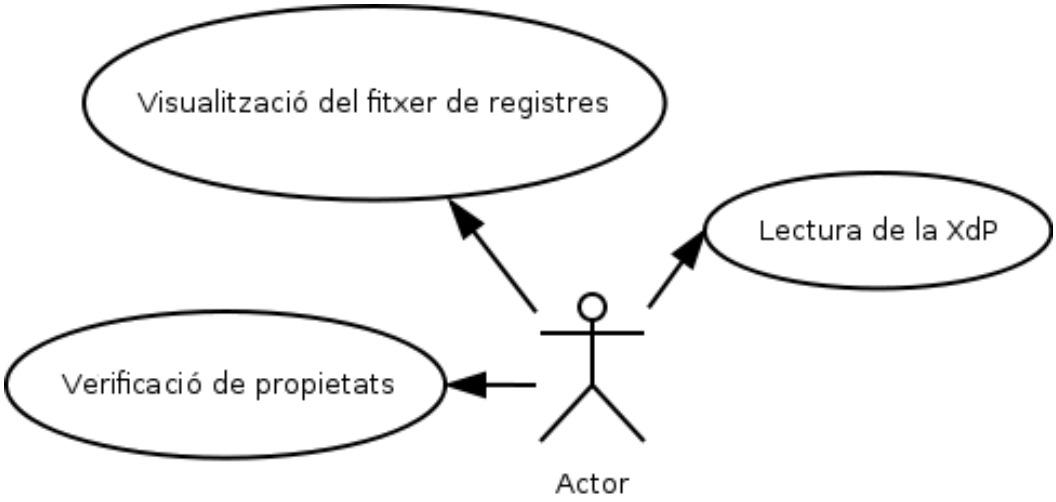


Figura 26: Casos d'ús de l'eina amb interfície gràfica

3.2.2.2.2 Llibreria estàtica

3.2.2.2.2.1 *Mòdul de XdP*

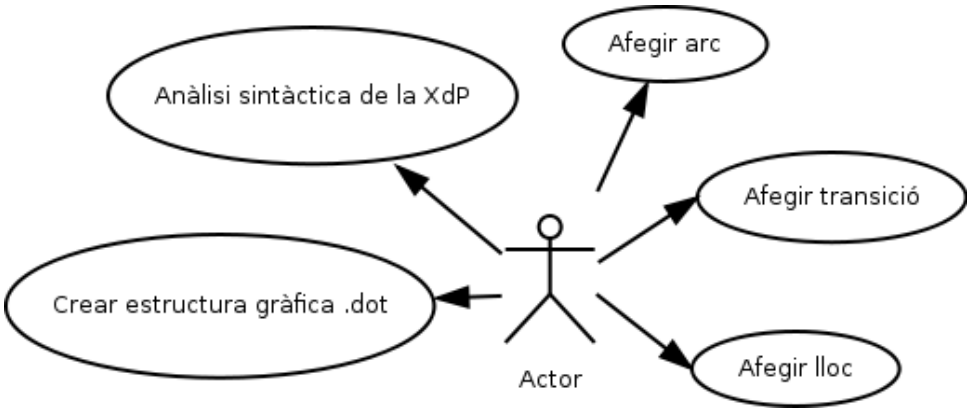


Figura 27: Casos d'ús del Mòdul de XdP

3.2.2.2.2.2 *Mòdul de PL*

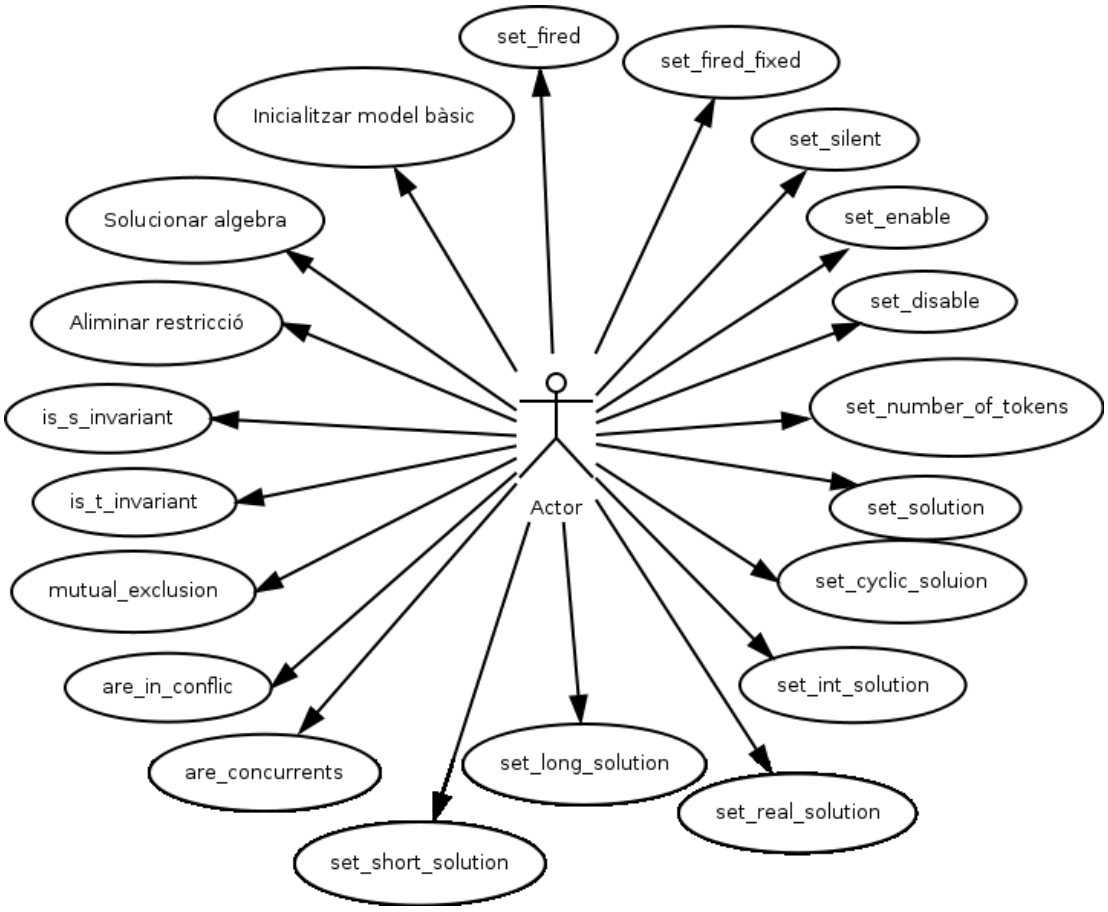


Figura 28: Casos d'ús del Mòdul de PL

3.2.2.2.2.3 *Mòdul de Propietats*

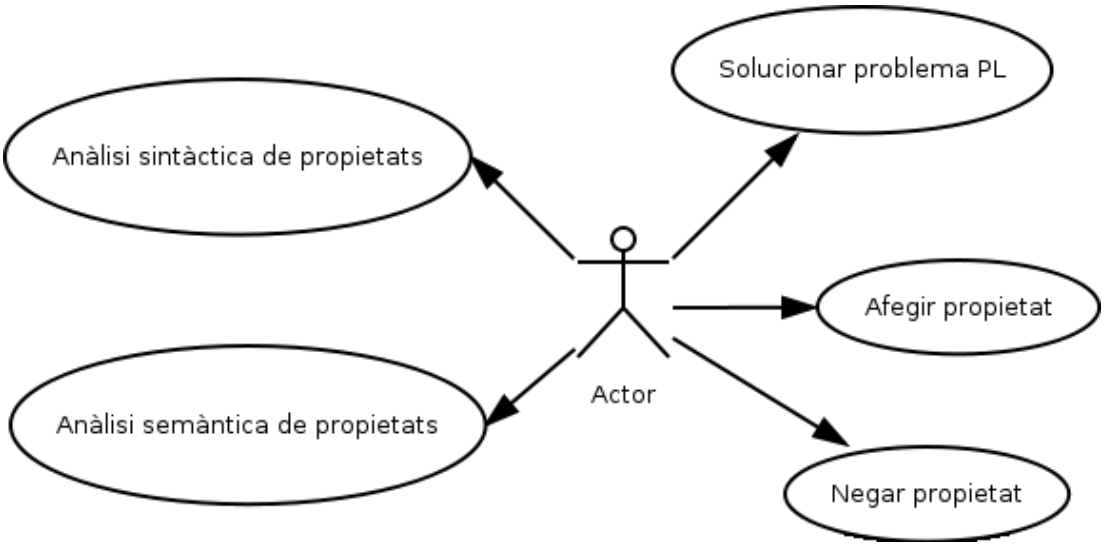


Figura 29: Casos d'ús del Mòdul de Propietats

3.2.2.3 Especificació dels casos d'ús

Al igual que pels diagrames de casos d'ús, s'ha separat l'especificació segons l'eina (interfície gràfica o llibreria) i segons el mòdul si s'escau.

3.2.2.3.1 Casos d'ús per a l'eina amb interfície gràfica

3.2.2.3.1.1 *Cas d'ús "Lectura de la XdP"*

Lectura de la XdP	
Actors	Analitzador.
Precondicions	-
Descripció	El sistema valida que el fitxer d'entrada contingui un XML ben format i que compleixi les restriccions que s'ha definit amb XML Schema. Crea una imatge amb la informació llegida sobre la XdP i la carrega a la interfície.
Postcondicions	S'ha creat una estructura de dades amb la informació de la XdP i s'ha carregat una imatge a la interfície gràfica amb la xarxa llegida.
Esdeveniments	1. L'analitzador selecciona el fitxer .xml que vol analitzar. 2. El sistema analitza sintàcticament el fitxer segons l'esquema .xsd definit per aquesta aplicació. 3. El sistema imprimeix al fitxer de registres el resultat de l'anàlisi sintàctica de la xarxa. 4. El sistema emmagatzema la informació llegida a una estructura de dades. 5. El sistema crea un fitxer .dot a partir de la XdP. 6. El sistema crea una imatge .png a partir del fitxer .dot. 7. El sistema carrega a la interfície la imatge generada.
Variants	4. El sistema carrega a la interfície un missatge amb l'error generat per l'anàlisi sintàctica de la xarxa.

3.2.2.3.1.2 *Cas d'ús "Verificació de Propietats"*

Verificació de Propietats	
Actors	Analitzador.
Precondicions	S'ha carregat una XdP.
Descripció	El sistema verifica les propietats introduïdes per l'usuari sobre la XdP escollida cridant un solucionador de problemes de PL.
Postcondicions	S'ha validat un conjunt de propietats sobre la XdP introduïda prèviament.

Esdeveniments	1. L'analitzador introdueix, en text pla, un conjunt de propietats. 2. El sistema analitza lèxica i sintàcticament el text introduït seguint una gramàtica definida per aquesta aplicació. 3. El sistema imprimeix al fitxer de registres el resultat de l'anàlisi sintàctica de les propietats. 4. El sistema crea una estructura de dades amb les propietats i realitza una optimització sobre elles per tal de reduir el nombre de crides al solucionador. 5. El sistema valida les propietats. 6. El sistema imprimeix al fitxer de registres el resultat de la validació de les propietats. 7. El sistema carrega a la interfície el resultat de la validació.
Variants	4. El sistema carrega a la interfície un missatge amb l'error generat per l'anàlisi lèxica o sintàctica.

3.2.2.3.1.3 *Cas d'ús "Visualització del fitxer de registres"*

Visualització del fitxer de registres	
Actors	Analitzador.
Precondicions	-
Descripció	El sistema mostra a l'usuari un fitxer amb els registres que l'aplicació ha generat.
Postcondicions	S'ha mostrat a la interfície el fitxer de registres de l'aplicació.
Esdeveniments	1. L'analitzador pitja el botó corresponent a la visualització del fitxer de registres. 2. El sistema mostra a la interfície el fitxer de registres.
Variants	4. El sistema indica a l'usuari que el fitxer de registres no existeix.

3.2.2.3.2 Casos d'ús per a la llibreria estàtica

3.2.2.3.2.1 *Mòdul de XdP*

3.2.2.3.2.1.1 Cas d'ús "Anàlisi sintàctica de la XdP"

Anàlisi sintàctica de la XdP	
Actors	Analitzador / Programador
Precondicions	-
Descripció	El sistema valida que el fitxer d'entrada contingui un XML

	ben format en base a un esquema definit amb XMLSchema.
Postcondicions	S'ha fet l'anàlisi sintàctica del fitxer.
Esdeveniments	1. L'usuari selecciona el fitxer que vol analitzar. 2. El sistema analitza sintàcticament el fitxer segons l'esquema definit per aquesta aplicació. 3. El sistema imprimeix al fitxer de registres el resultat de l'anàlisi sintàctica de la xarxa. 4. El sistema crea una estructura de dades que conté la XdP. 5. El sistema retorna un booleà indicant l'èxit de la operació.
Variants	4. El sistema retorna un booleà indicant que la operació ha fallat.

3.2.2.3.2.1.2 Cas d'ús "Crear estructura gràfica .dot"

Crear estructura gràfica .dot	
Actors	Analitzador / Programador
Precondicions	
Descripció	El sistema genera un fitxer en llenguatge DOT que defineix l'estructura d'una XdP donada i com ha de ser dibuixada.
Postcondicions	S'ha generat un fitxer .dot que defineix la XdP.
Esdeveniments	1. L'usuari selecciona la XdP de la que vol generar l'esquema gràfic en format textual. 2. El sistema genera un fitxer que conté l'estructura de la XdP i instruccions sobre el disseny del graf de la xarxa.

3.2.2.3.2.1.3 Cas d'ús "Afegir lloc"

Afegir lloc	
Actors	Analitzador / Programador
Precondicions	Existeix una estructura de dades que conté una XdP vàlida (compleix l'estructura definida amb XMLSchema per l'aplicació) o que està buida.
Descripció	El sistema afegeix a la XdP donada un lloc amb els atributs i les relacions indicats per l'usuari.
Postcondicions	S'ha modificat la XdP afegint un nou lloc i les seves relacions.
Esdeveniments	1. L'usuari indica l'identificador, el nom i el nombre de marques del lloc, així com les possibles relacions amb transicions, tant de pre-conjunt com de post-conjunt.

	2. El sistema crea un nou lloc amb els atributs . 3. El sistema afegeix la nova transició a la XdP amb les relacions indicades.

3.2.2.3.2.1.4 Cas d'ús "Afegir transició"

Afegir transició	
Actors	Analitzador / Programador
Precondicions	Existeix una estructura de dades que conté una XdP vàlida (compleix l'estructura definida amb XMLSchema per l'aplicació) o que està buida.
Descripció	El sistema afegeix a una XdP donada una transició amb els atributs i les relacions indicats per l'usuari.
Postcondicions	S'ha modificat la XdP afegint una nova transició.
Esdeveniments	1. L'usuari indica l'identificador i el nom de la transició, així com les possibles relacions amb llocs, tant de pre-conjunt com de post-conjunt. 2. El sistema crea una nova transició amb els atributs indicats. 3. El sistema afegeix la nova transició a la XdP amb les relacions indicades.

3.2.2.3.2.1.5 Cas d'ús "Afegir arc"

Afegir arc	
Actors	Analitzador / Programador
Precondicions	Existeix una estructura de dades que conté una XdP vàlida (compleix l'estructura definida amb XMLSchema per l'aplicació) o que està buida. Existeixen l'origen i el destí de l'arc que es vol afegir.
Descripció	El sistema afegeix a una XdP donada un arc amb els atributs i les relacions indicades per l'usuari.
Postcondicions	S'ha modificat la XdP afegint un nou arc.
Esdeveniments	1. L'usuari indica l'identificador, el nom i l'origen i destí de l'arc en forma d'identificadors. 2. El sistema crea un nou arc amb els atributs indicats. 3. El sistema afegeix el nou arc a la XdP i modifica l'origen i el destí amb el seu nou component al post-conjunt i al pre-conjunt, respectivament.

3.2.2.3.2.2 *Mòdul de PL*

3.2.2.3.2.2.1 Cas d'ús "Inicialitzar model bàsic"

Inicialitzar model bàsic	
Actors	Analitzador / Programador
Precondicions	Existeix un XdP vàlida.
Descripció	El sistema inicia els atributs bàsics de l'àlgebra que contindrà el model de PL i crea el model. S'afegeix la restricció per defecte.
Postcondicions	S'ha inicialitzat el model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica que vol inicial el model de PL. 2. El sistema inicialitza els valors bàsics de l'àlgebra a partir de la informació de la XdP: nombre de llocs, nombre de transicions, nombre de traces i matriu d'incidència. 3. El sistema crea un solucionador de problemes de PL amb tantes variables com transicions tingui la XdP. 4. El sistema introdueix com a restricció per defecte que els valors de la solució per les variables del problema han de ser nombres positius o iguals a zero, és a dir, que el model ha de complir $Ax >= -M_0$.

3.2.2.3.2.2.2 Cas d'ús "Disparar transició"

Disparar transició	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix al model de PL una restricció que representa el dispar d'una transició concreta a una traça.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica quina transició vol que es dispari (un nombre indefinit de vegades, més gran que zero) i per quina traça ho farà. 2. El sistema genera la restricció sobre la traça indicada i l'afegeix al model de PL.

3.2.2.3.2.2.3 Cas d'ús "Disparar transició N vegades"

Disparar transició N vegades	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.

Descripció	El sistema afegeix a un model de PL una restricció que representa el dispar d'una transició concreta, un nombre definit de vegades, per una traça indicada.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica quina transició vol que es dispari, quantes vegades ho ha de fer i per quina traça s'haurà d'aplicar la restricció. 2. El sistema genera la restricció corresponent i l'afegeix al model de PL.

3.2.2.3.2.2.4 Cas d'ús "Silenciar transició"

Silenciar transició	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix a un model de PL una restricció que representa que una transició concreta no s'ha de disparar per una traça indicada.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica quina transició vol que no es dispari i a quina traça s'ha d'aplicar la restricció. 2. El sistema genera la restricció i l'afegeix al model de PL.

3.2.2.3.2.2.5 Cas d'ús " És S-invariant"

És S-invariant	
Actors	Analitzador / Programador
Precondicions	S'ha creat i inicialitzat un objecte d'àlgebra lineal.
Descripció	El sistema comprova si un vector donat és un S-invariant per a una XdP específica.
Postcondicions	S'ha comprovat la restricció aplicant àlgebra lineal.
Esdeveniments	1. L'usuari dóna un vector *x* amb tantes posicions com llocs tingui la XdP on s'aplica la restricció. 2. El sistema comprova que es compleixi l'equació $xA = 0$. 3. El sistema retorna un booleà indicant si el vector introduït és o no un S-invariant del model.

3.2.2.3.2.2.6 Cas d'ús "És T-invariant"

És T-invariant	
Actors	Analitzador / Programador

Precondicions	S'ha creat i inicialitzat un objecte d'àlgebra lineal.
Descripció	El sistema comprova si un vector donat és un T-invariant per a una XdP específica.
Postcondicions	S'ha comprovat la restricció sobre la XdP aplicant àlgebra lineal.
Esdeveniments	1. L'usuari dóna un vector *x* amb tantes posicions com transicions tingui la XdP on s'aplica la restricció. 2. El sistema comprova que es compleixi l'equació *Ax = 0*. 3. El sistema retorna un booleà indicant si el vector introduït és o no un T-invariant del model.

3.2.2.3.2.2.7 Cas d'ús "Hi ha conflicte"

Hi ha conflicte	
Actors	Analitzador / Programador
Precondicions	S'ha creat i inicialitzat un objecte d'àlgebra lineal.
Descripció	El sistema comprova si hi ha conflicte per dues transicions concretes després de disparar un conjunt de transicions determinat.
Postcondicions	S'ha comprovat la restricció sobre la XdP aplicant àlgebra lineal.
Esdeveniments	1. L'usuari introdueix un vector de dispars i dues transicions. 2. El sistema comprova que les dues transicions indicades estiguin actives després d'executar els dispars indicats. 3. El sistema comprova que $\lvert \cdot y \rvert + \lvert \cdot z \rvert > \sum_{p \in \cdot y U \cdot z} M[p]$. 4. El sistema retorna un enter indicant si hi ha o no conflicte i per què.
Variants	3. El sistema retorna un booleà indicant que no hi ha conflicte i per què.

3.2.2.3.2.2.8 Cas d'ús "Hi ha concurrència"

Hi ha concurrència	
Actors	Analitzador / Programador
Precondicions	S'ha creat i inicialitzat un objecte d'àlgebra lineal.
Descripció	El sistema comprova si hi ha concurrència per dues transicions concretes després de disparar una seqüència determinada de transicions.
Postcondicions	S'ha comprovat la restricció aplicant àlgebra lineal.

Esdeveniments	1. L'usuari introdueix un vector de dispars i dues transicions. 2. El sistema comprova que les dues transicions indicades estiguin actives després d'executar els dispars indicats. 3. El sistema comprova que $\lvert \cdot y \rvert + \lvert \cdot z \rvert \leq \sum_{p \in \cdot y U \cdot z} M[p]$. 4. El sistema retorna un enter indicant si hi ha o no concurrència i per què.
	3. El sistema retorna un booleà indicant que no hi ha concurrència i per què.

3.2.2.3.2.2.9 Cas d'ús "Exclusió mútua"

Exclusió mútua	
Actors	Analitzador / Programador
Precondicions	-
Descripció	El sistema crea un nou model de PL i afegeix la restricció d'exclusió mútua per a dos llocs d'una XdP.
Postcondicions	S'ha solucionat la restricció amb el solucionador de PL.
Esdeveniments	1. L'usuari indica els dos llocs, p_1 i p_2, pels quals vol comprovar si hi ha exclusió mútua. 2. El sistema crea un model de PL i comprova si existeix alguna solució en que estiguin marcats tant p_1 com p_2. 3. El sistema retorna un booleà que indica si s'ha complert o no la restricció.

3.2.2.3.2.2.10 Cas d'ús "Transició activa"

Transició activa	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix al model de PL una restricció que representa l'activació d'una transició concreta per a una traça específica.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	4. L'usuari indica quina transició vol que estigui activa i al final de l'execució de quina traça ho ha d'estar. 5. El sistema genera la restricció corresponent fent que cada lloc del pre-conjunt de la transició tingui, com a mínim una marca al final de l'execució [$M_{0i} + Ax >= 1$ on i és cada posició dels llocs del pre-conjunt de la transició]. El sistema afegeix al model de PL.

3.2.2.3.2.2.11 Cas d'ús "Transició inactiva"

Transició inactiva	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix al model de PL una restricció que representa la desactivació d'una transició concreta per a una traça específica.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica quina transició vol que estigui inactiva i al final de l'execució de quina traça ho ha d'estar. 2. El sistema genera la restricció corresponent fent que al menys un lloc del pre-conjunt de la transició estigui buit al final de l'execució [$M_{0i} + Ax = 0$ on i és cada posició dels llocs del pre-conjunt de la transició]. El sistema afegeix al model de PL.

3.2.2.3.2.2.12 Cas d'ús "Nombre de marques"

Nombre de marques	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix al model de PL una restricció que representa el nombre de marques que ha de tenir un lloc concret per una traça específica.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica el nombre de marques nt, el lloc i i la traça que vol restringir. 2. El sistema genera la restricció corresponent [$Ax_i = nt - M_{0i}$] i l'afegeix al model de PL.

3.2.2.3.2.2.13 Cas d'ús "Solució abastable"

Solució abastable	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix al model de PL una restricció que representa el marcatge que ha de tenir la XdP al final de l'execució d'una traça determinada.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.

Esdeveniments	1. L'usuari introdueix un vector M_p, amb tantes posicions com llocs té la XdP que es vol restringir; aquest vector indica el nombre de marques que ha de tenir cada lloc de la xarxa, i la traça al final de la qual s'ha de complir la restricció. 2. El sistema genera la restricció corresponent [$M_0 + Ax = M_p$] i l'afegeix al model de PL.

3.2.2.3.2.2.14 Cas d'ús "Solució cíclica"

Solució cíclica	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix al model de PL una restricció que verifica si la XdP és cíclica per un traça concreta.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari introdueix la traça per la qual vol comprovar si la XdP és cíclica. 2. El sistema genera la restricció corresponent [$Ax = 0$] i l'afegeix al model de PL.

3.2.2.3.2.2.15 Cas d'ús "Solució entera"

Solució entera	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix a un model de PL una restricció que obliga que la solució per una traça donada de totes les variables siguin nombre enters.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari introdueix la traça que vol restringir per tenir una solució dintre del rang del enters. 2. El sistema genera la restricció corresponent, fent que totes les variables de la traça indicada siguin valors enters, i l'afegeix al model de PL.

3.2.2.3.2.2.16 Cas d'ús "Solució real"

Solució real	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.

Descripció	El sistema afegeix a un model de PL una restricció que permet que la solució per una traça donada de totes les variables siguin nombre real.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari introdueix la traça que vol restringir per tenir una solució dintre del rang del reals. 2. El sistema genera la restricció corresponent, fent que totes les variables de la traça indicada siguin valors reals, i l'afegeix al model de PL.

3.2.2.3.2.2.17 Cas d'ús "Solució curta"

Solució curta	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix a un model de PL una restricció que obliga a que la solució de totes les variables del problema per una traça donada sigui binària, és a dir, tingui valors entre zero i un. S'anomena solució curta per què es restringeix notablement el nombre de valors que pot tenir cada variable.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari introdueix la traça que vol restringir per tenir solucions curtes. 2. El sistema genera la restricció corresponent, fent que totes les variables de la traça indicada tinguin valors entre 0 i 1, i l'afegeix al model de PL.

3.2.2.3.2.2.18 Cas d'ús "Solució llarga"

Solució llarga	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema afegeix a un model de PL una restricció que permet la solució de qualsevol de les variables del problema per una traça donada sigui un valor de coma flotant. S'anomena solució llarga per què el nombre de possibles valors de les variables no està limitat.
Postcondicions	S'ha afegit la restricció al model d'àlgebra lineal.
Esdeveniments	1. L'usuari introdueix la traça que vol restringir per permetre solucions llargues. 2. El sistema genera la restricció corresponent, permetent

	que totes les variables de la traça indicada tinguin qualsevol valor, i l'afegeix al model de PL.

3.2.2.3.2.2.19 Cas d'ús "Eliminar restricció"

Eliminar restricció	
Actors	Analitzador / Programador
Precondicions	La restricció que es vol eliminar existeix al solucionador.
Descripció	El sistema elimina la restricció indicada per l'usuari del model de PL.
Postcondicions	S'ha eliminat la restricció del model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica la restricció que vol eliminar. 2. El sistema elimina la restricció del model de PL. Es tracten de forma separada les restriccions de columna (s'apliquen concretament a una variable) i les restriccions de fila (afecten a totes les variables).

3.2.2.3.2.2.20 Cas d'ús "Solucionar problema"

Solucionar problema	
Actors	Analitzador / Programador
Precondicions	S'ha creat un solucionador de PL.
Descripció	El sistema soluciona el problema de PL i retorna a l'usuari el resultat obtingut.
Postcondicions	S'ha intentat solucionar el model d'àlgebra lineal.
Esdeveniments	1. L'usuari indica que vol solucionar el model de PL creat. 2. El sistema crida al solucionador de problemes i recull la solució. 3. El sistema imprimeix al fitxer de registres el resultat de la verificació de les propietats obtinguda pel solucionador. 4. El sistema retorna un booleà indicant l'èxit o el fracàs de la operació.

3.2.2.3.2.3 *Mòdul de Propietats*

3.2.2.3.2.3.1 Cas d'ús "Anàlisi sintàctica de Propietats"

Anàlisi sintàctica de propietats	
Actors	Analitzador / Programador
Precondicions	-

Descripció	El sistema llegeix un text introduït per l'usuari i l'analitza lèxica i sintàcticament a partir de la gramàtica definida per aquesta aplicació.
Postcondicions	S'ha fet l'anàlisi lèxica i sintàctica de la cadena d'entrada.
Esdeveniments	1. L'usuari introdueix una cadena de text amb les propietats que vol analitzar. 2. El sistema analitza lèxica i sintàcticament la cadena en base a una gramàtica definida específicament pel tipus de propietats que es vol validar. 3. El sistema imprimeix al fitxer de registres el resultat de l'anàlisi sintàctica de les propietats. 4. El sistema retorna un booleà indicant l'èxit o el fracàs de la operació.

3.2.2.3.2.3.2 Cas d'ús "Anàlisi semàntica de Propietats"

Anàlisi semàntica de propietats	
Actors	Analitzador / Programador
Precondicions	La cadena d'entrada és sintàcticament correcta.
Descripció	El sistema analitza semànticament la cadena d'entrada segons les propietats definides en l'aplicació i les seves característiques i crea una estructura de dades que conté aquestes propietats.
Postcondicions	S'ha fet l'anàlisi semàntica de la cadena d'entrada i s'ha creat una estructura de dades amb les propietats llegides.
Esdeveniments	1. L'usuari introdueix una cadena de text amb les propietats que vol analitzar. 2. El sistema analitza semànticament la cadena identificant els diferents tipus d'elements (propietats, paràmetres i operadors) i comprova que les dades són vàlides (pertanyen a la xarxa seleccionada) i estan ben construïdes (els paràmetres són els correctes per a cada propietat, tant en nombre com en definició). 3. El sistema imprimeix al fitxer de registres el resultat de l'anàlisi semàntica de les propietats. 4. El sistema crea una estructura de dades que conté les propietats d'entrada amb els seus paràmetres corresponents i els operadors que les relacionen. 5. El sistema retorna un booleà indicant l'èxit de la operació.
Variants	4. El sistema retorna un booleà indicant el fracàs de la operació.

3.2.2.3.2.3.3 Cas d'ús "Negar propietat"

Negar propietat	
Actors	Analitzador / Programador
Precondicions	Existeix una estructura de propietats amb, al menys, una propietat.
Descripció	El sistema modifica l'estructura canviat, si pot, la propietat indicada per la seva negació.
Postcondicions	El sistema ha negat una propietat de l'estructura.
Esdeveniments	1. L'usuari indica la propietat que vol negar. 2. El sistema modifica la propietat indicada per la propietat que representa la seva negació. 3. El sistema retorna un booleà indicant l'èxit de la operació.
Variants	2. El sistema retorna un booleà indicant el fracàs de la operació (la propietat indicada no es pot negar).

3.2.2.3.2.3.4 Cas d'ús "Solucionar problema"

Solucionar problema	
Actors	Analitzador / Programador
Precondicions	Existeix una estructura de propietats amb, al menys, una propietat.
Descripció	El sistema crea un model de programació lineal a partir d'unes propietats predefinides i el soluciona.
Postcondicions	El sistema ha verificat un conjunt de propietats mitjançant un solucionador de problemes de PL.
Esdeveniments	1. L'usuari indica que vol verificar un conjunt de propietats predefinit en una estructura de dades. 2. El sistema genera restriccions per a un problema de PL mitjançant les va llegint d'una estructura de dades que conté propietats sobre una XdP. Per cada node mínim d'execució que es troba (màxim conjunt de propietats que tenen la mateixa prioritat i és màxima) el sistema crea un model d'àlgebra lineal i el soluciona. 3. El sistema imprimeix al fitxer de registres el resultat de solucionar el problema d'àlgebra lineal. 4. El sistema retorna un booleà amb el resultat.

3.2.3 Interfície

S'ha desenvolupat una interfície gràfica per tal de facilitar l'ús de l'eina desenvolupada. Aquesta interfície permet realitzar les operacions anteriorment definides amb els casos d'ús. A continuació es mostra l'aspecte general de la interfície i es detalla els seus components.

3.2.3.1 Components

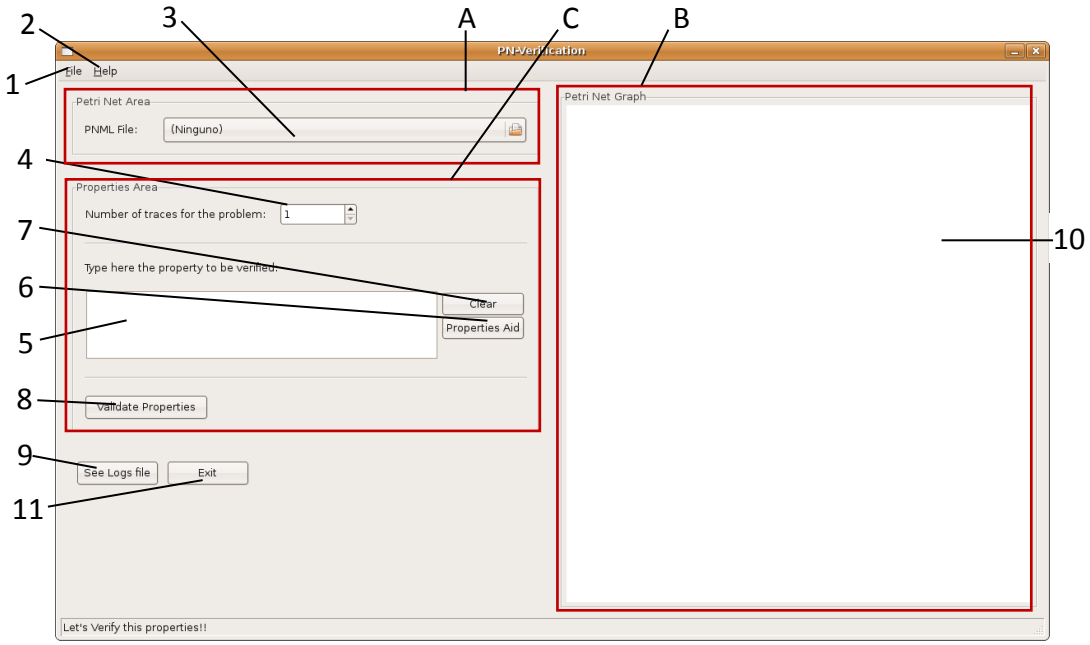


Figura 30: Esquema de la Interfície Gràfica de l'eina PN-Verification

Descripció general de les tres àrees de treball:

A. Àrea per escollir la XdP.
B. Àrea per visualitzar la XdP.
C. Àrea per verificar les propietats.

Descripció dels components de la interfície:

1. **File**

   Botó amb les funcions generals del programa (Exit).

2. **Help**

   Botó amb una petita introducció sobre el programa.

3. **PNML File**

   Navegador de directoris que permet escollir fitxers en format pnml que tinguin extensió .pnml o .xml.

   Aquest camp serveix per carregar un fitxer que descrigui una XdP.

4. **Number of traces for the problem**

   Comptador amb valor mínim igual a 1.

   Aquest camp permet escollir el nombre de traces que tindrà el problema que volem solucionar.

5. **Type here the property to be verified**

   Camp de text que permet entrar valors alfanumèrics.

   Aquest camp s'usa per introduir les propietats que es vol verificar.

6. **Properties Aid**

   Botó que obre una finestra amb una petita ajuda sobre les propietats que es permet verificar i com s'ha d'escriure-les.

7. **Clear**

   Botó que neteja l'entrada del Camp de text 5.

8. **Validate properties**

   Botó que s'activa quan s'ha escollit una XdP al Camp 3 i s'ha introduït algun valor al Camp 5. Permet verificar les propietats sobre la XdP.

9. **See logs file**

   Botó que carrega el fitxer de registres de l'aplicació. Si el fitxer no existeix mostra un missatge indicant aquesta situació.

10. **Petri Net graph**

    Area on, una vegada carregada una XdP vàlida al Camp 3, es mostra el graf corresponent a la XdP.

11. **Exit**

    Botó que permet sortir de l'aplicació.

4 Disseny

La part de disseny del programari dins de l'enginyeria del software consisteix a establir una estratègia de solució a partir dels requisits especificats anteriorment. En aquesta solució s'ha de definir el sistema amb el detall suficient com per que un programador prèviament desconeixedor del sistema sigui capaç de desenvolupar-lo físicament.

Per al desenvolupament d'aquesta part es tindrà en compte els requisits no funcionals definits durant l'especificació i s'analitzarà les diferents alternatives tecnològiques tot justificant les decisions preses per a cada cas (solucions que seran usades durant la implementació). En aquest punt es parla de patrons de disseny i es mostra l'arquitectura, els components i els diagrames de seqüència del sistema.

4.1 Definició de l'arquitectura

S'ha usat una arquitectura per capes amb l'objectiu de separar la lògica del sistema de la lògica del disseny:

- **Capa de presentació**

  És la part de l'aplicació que conté la interfície gràfica i tota la lògica que permet la comunicació entre l'usuari i el programa. La seva funció principal és rebre tasques per part de l'usuari i reportar-li els resultats del programa d'una forma intel·ligible.

- **Capa de domini**

  Aquest nivell té com a missió principal la coordinació de l'aplicació. S'encarrega de prendre decisions i realitzar càlculs a partir de les peticions de la capa de presentació i de processar aquesta informació cap a les dues capes contigües.

  A la capa de domini s'ha desenvolupat un sistema de **control d'errors** que es recullen de les diferents llibreries utilitzades i que poden venir donats únicament per errors detectats en les entrades. Aquests errors són degudament tractats en aquesta capa per tal d'indicar a l'usuari què ha passat d'una manera intel·ligible per ell i deixen els missatges corresponents al fitxer de registres.

- **Capa de dades**

  Aquesta capa representa tant el lloc on resideixen les dades com la part de l'aplicació que s'encarrega d'accedir-hi.

L'aplicació desenvolupada no requereix d'una Base de Dades ja que tots els càlculs es fan de forma dinàmica. Les úniques dades permanents són les informacions que queden registrades sobre les operacions executades i que es guarden en un fitxer de registres emmagatzemat a la memòria secundària de l'ordinador.

Les capes de domini i dades són iguals per les dues eines desenvolupades. La capa de presentació pertany únicament a l'aplicació, anomenada **PN_Verification**.

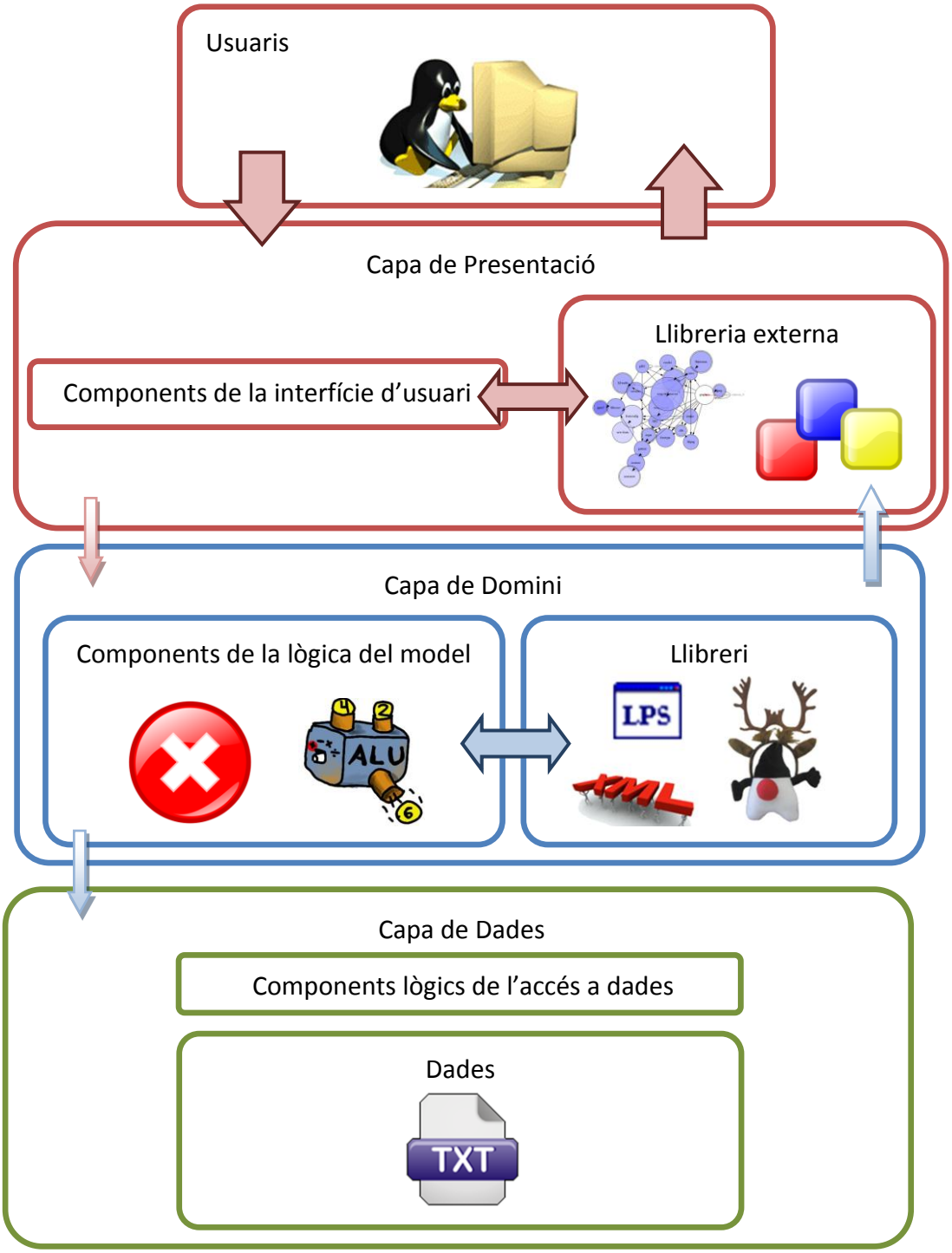


Figura 31: Arquitectura del sistema

4.2 Diagrames de classe

Els diagrames de classe presentats a continuació deriven dels models de dades definits a la part d'especificació (apartat 3.2.1 d'aquest document). En aquest cas el nivell de detall és major i s'especifica detalls com les cardinalitats i les funcions per a tots els casos d'ús, així com les restriccions textuals si s'escau.

El primer que es mostra és l'esquema de dades general, que conté totes les classes del sistema, sense atributs ni operadors, per tal de fer-lo més llegible:

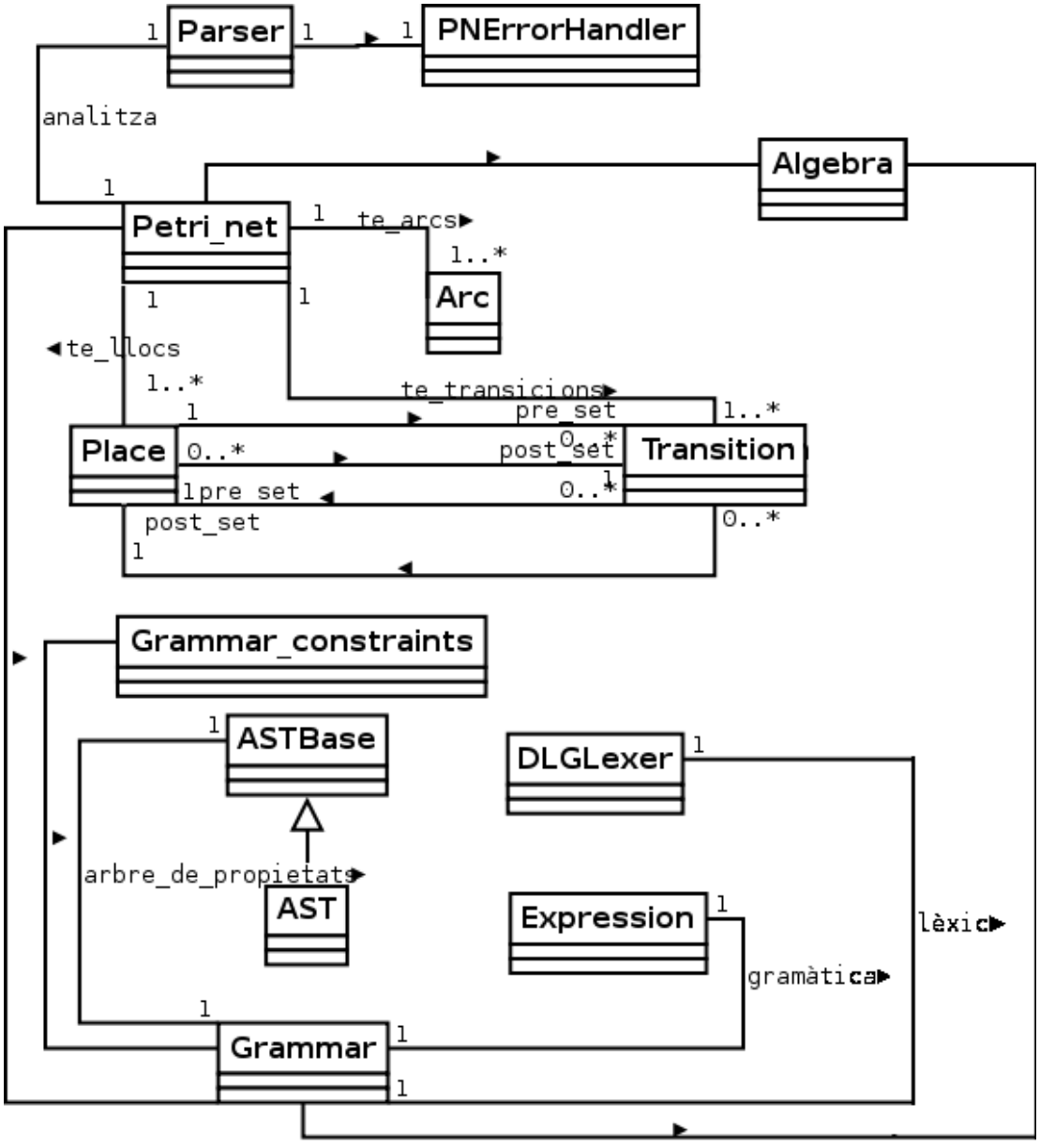


Figura 32: Diagrama de classes general

Als següents punts es mostra amb tot detall l'estructura de dades de cada mòdul. No apareix el mòdul de PL per que resta igual que a l'especificació (apartat 3.2.1.2 d'aquest document).

4.2.1 Mòdul de XdP

A continuació s'especifica tots els objectes del mòdul de XdP en un mateix diagrama:

- Estructura de la XdP
- Analitzador sintàctic de la XdP
- Relacions entre ells

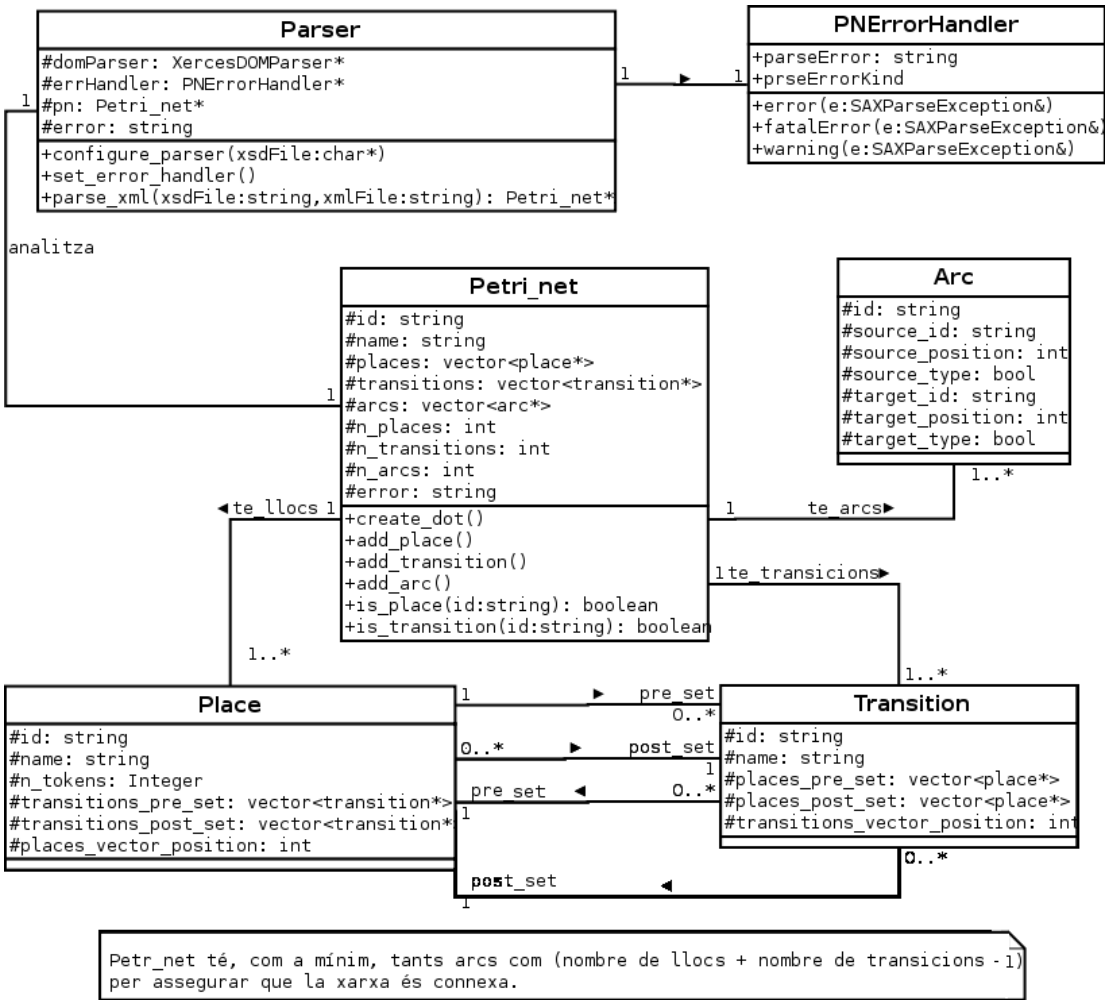


Figura 33: Diagrama de classes del mòdul de XdP

4.2.2 Mòdul de Propietats

Es mostra en el mateix diagrama:

- Estructura de dades que conté les propietats a verificar sobre una XdP en l'ordre en què les ha introduït l'usuari.
- Components que permeten fer l'anàlisi sintàctica i semàntica de les propietats, i cridar a un model de PL per solucionar el problema.

Figura 34: Diagrama de classes del màdul de Propietats

4.3 Diagrames de seqüència

En aquest apartat es mostra els diagrames de seqüència dels mètodes més significatius del sistema per les dues eines desenvolupades.

Per facilitar la lectura i mostrar la modularitat del sistema, es separa cada diagrama de cas d'ús en petits sub-diagrames en funció de l'objecte que tingui la responsabilitat de realitzar cada tasca.

4.3.1 Diagrames per a l'eina amb interfície gràfica

4.3.1.1 Diagrama de seqüència de "Lectura de la XdP"

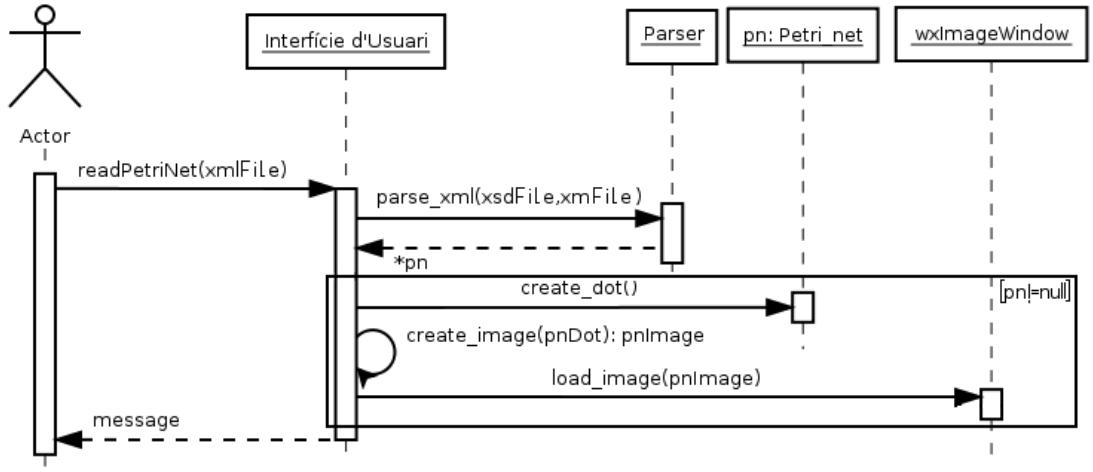


Figura 35: Funció de lectura de la XdP

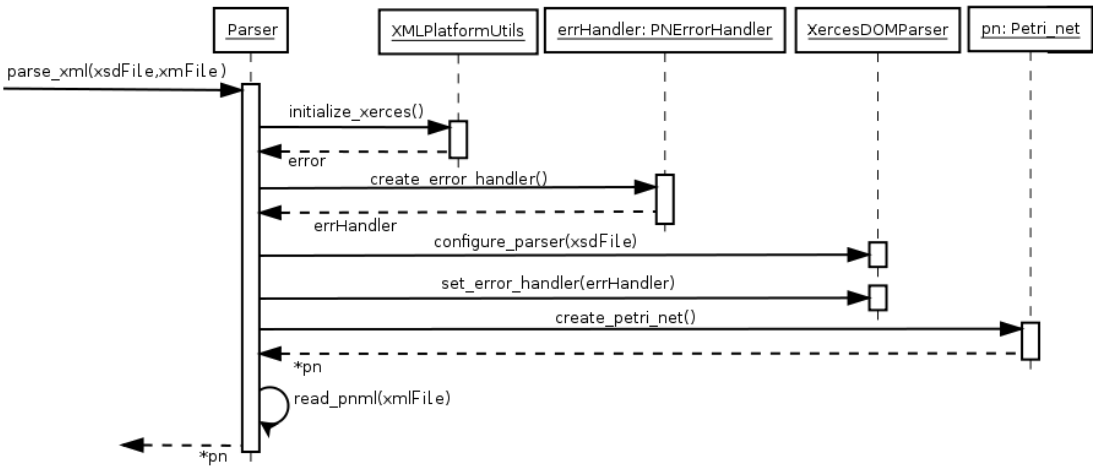


Figura 36: Funció d'anàlisi sintàctica de la XdP

4.3.1.2 Diagrama de seqüència de "Verificació de Propietats"

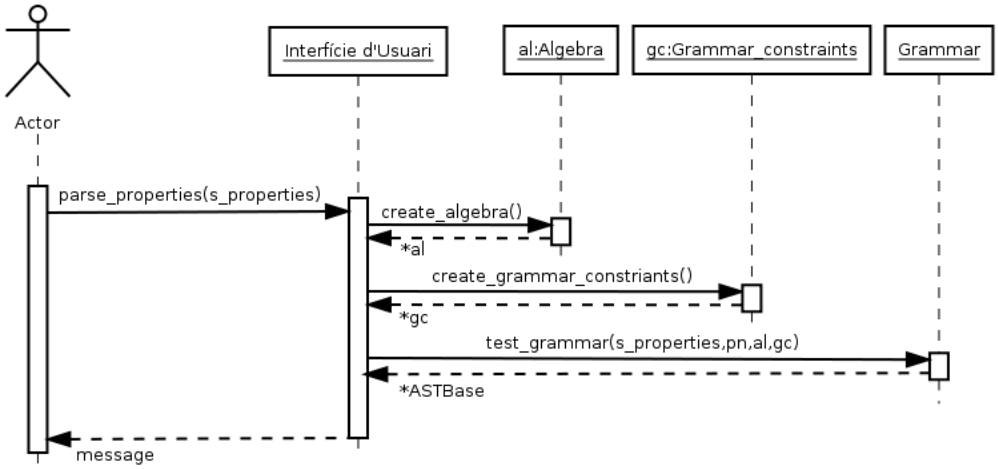


Figura 37: Funció de verificació de les propietats

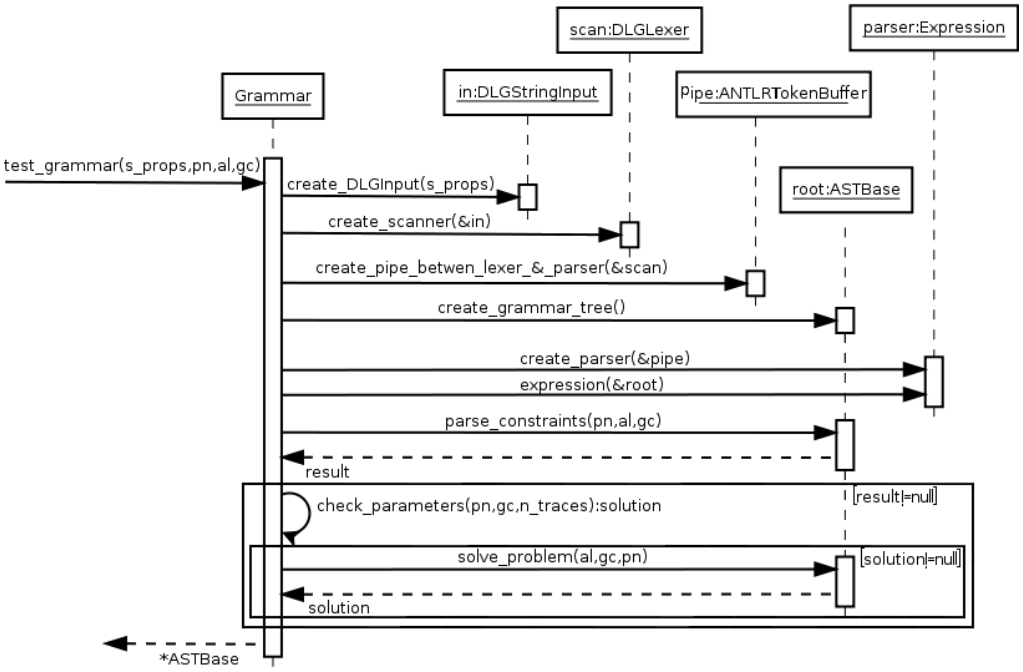


Figura 38: Figura per l'anàlisi i la verificació de propietats

La funció **solve_problem** s'encarrega de recórrer *gc*, l'estructura de dades amb les propietats, i anar configurant el model de *PL* que té *al* per solucionar el problema segons la prioritat de les operacions.

La funció *parse_constraints()* apareix al mòdul de Propietats (apartat 4.3.2.2 d'aquest document). La *check_parameters()* pertany també a aquest mòdul tot.

4.3.2 Diagrames per la llibreria estàtica

Per la llibreria estàtica s'especificarà els diagrames de les funcions més importants dels mòduls de PL i de Propietats. Pel mòdul de XdP la funció més significativa és l'anàlisi sintàctica de la XdP, i ja s'ha especificat als diagrames de l'eina amb interfície. Es considera trivials les funcions que creen i modifiquen les dades d'una XdP.

4.3.2.1 Mòdul de PL

S'ha extret del model els tipus de propietats que es pot verificar i s'ha agrupat segons:

- Funció que no requereix cridar al solucionador de PL per ser verificada:
  - Comprovar si dos transicions estan en conflicte o són concurrents.
  - Comprovar si un vector és un S-invariant o un T-invariant.
- Funció que afegeix restriccions de fila (es basen en l'equació de marcatge i afecten a totes les variables del problema):
  - La solució ha de ser positiva (restricció per defecte del problema).
  - La solució ha de ser una en concret.
  - La solució ha de ser cíclica.
  - Una transició ha de estar activa o inactiva al final de l'execució.
  - Un lloc ha de tenir un nombre determinat de fitxes al final de l'execució.
- Funció que afegeix restriccions de columna (restriccions sobre les variables):
  - La solució ha d'estar al domini dels enters o dels reals.
  - La solució ha de ser binària o no.
  - Una transició s'ha de disparar o no es pot disparar.
- Funció que genera un model auxiliar: comprovar si hi ha exclusió mútua entre dos llocs.

A continuació es mostra , a més del diagrama d'inicialització el model, tres diagrames a mode d'exemple, un per cada tipus de restricció.

4.3.2.1.1 Diagrama del cas d'ús "Inicialitzar model bàsic"

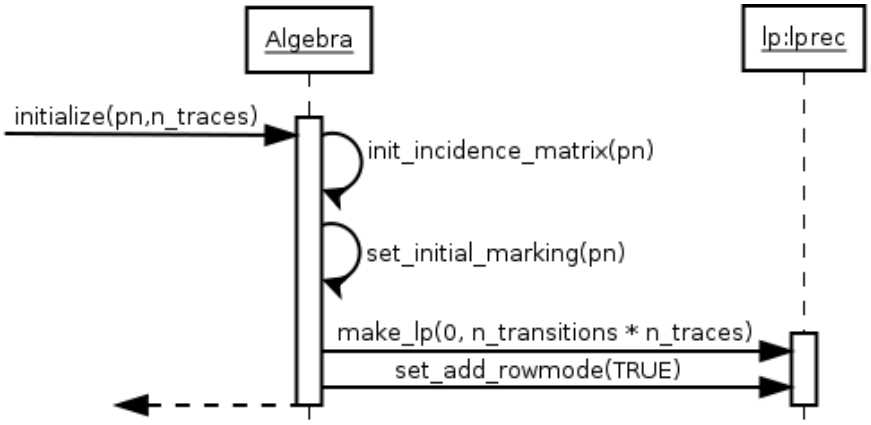


Figura 39: Diagrama per la inicialització del solucionador de PL

4.3.2.1.2 Diagrama del cas d'ús "Hi ha conflicte"

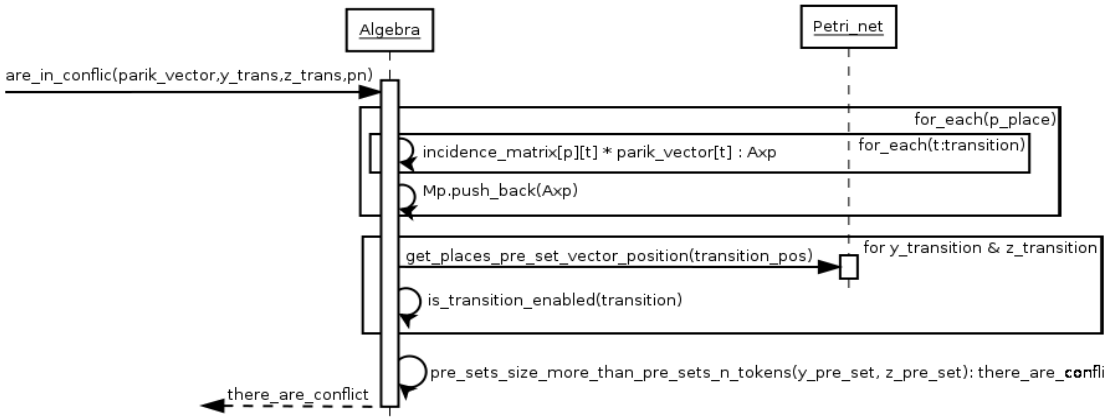


Figura 40: Diagrama de la funció que comprova la veracitat de la propietat "dos transicions estan en conflicte" (no crida al solucionador)

4.3.2.1.3 Diagrama del cas d'ús "Solució específica"

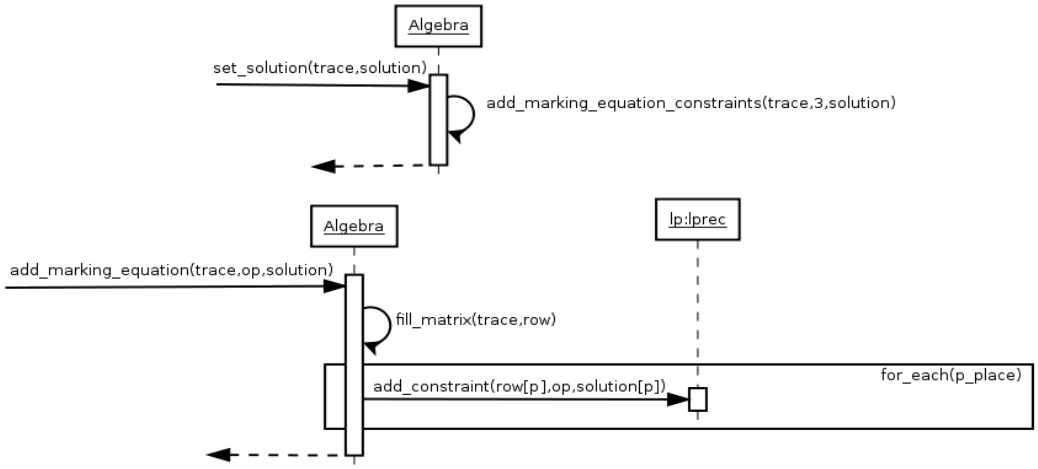


Figura 41: Diagrama de la funció que afegeix al model la propietat de restringir la solució a un marcatge final específic per una traça donada (restricció de fila)

4.3.2.1.4 Diagrama del cas d'ús "Solució entera"

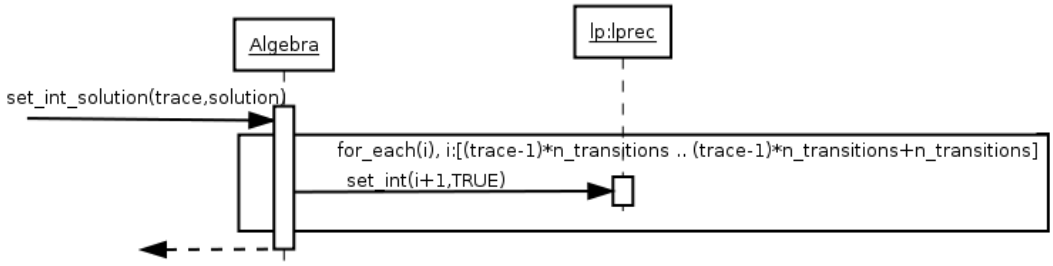


Figura 41: Diagrama de la funció que restringeix la solució del problema de forma que la solució ha de ser de tipus enter per una traça donada (restricció de columna)

4.3.2.2 Mòdul de Propietats

En aquest mòdul es mostra els diagrames de les funcions que realitzen la validació de les propietats i la solució del problema així com la funció que fa la negació d'una restricció de l'estructura de dades que conté les propietats.

4.3.2.2.1 Diagrama del cas d'ús "Anàlisi sintàctica de les Propietats"

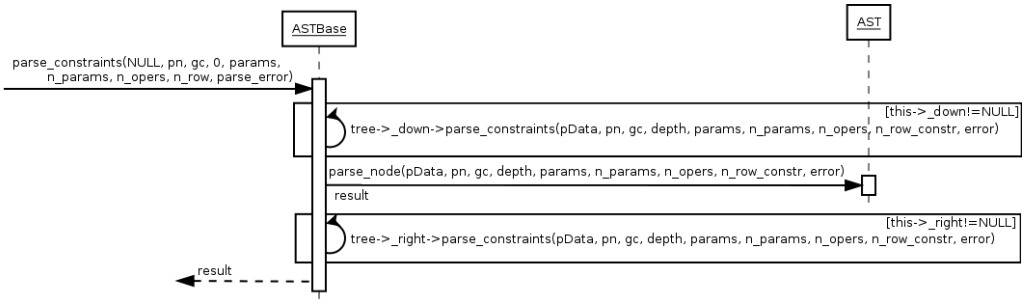


Figura 42: Diagrama de la funció que fa un recorregut en **inordre** de l'arbre generat per la gramàtica amb les propietats i crida a la funció que en fa l'anàlisi lèxica i sintàctica.

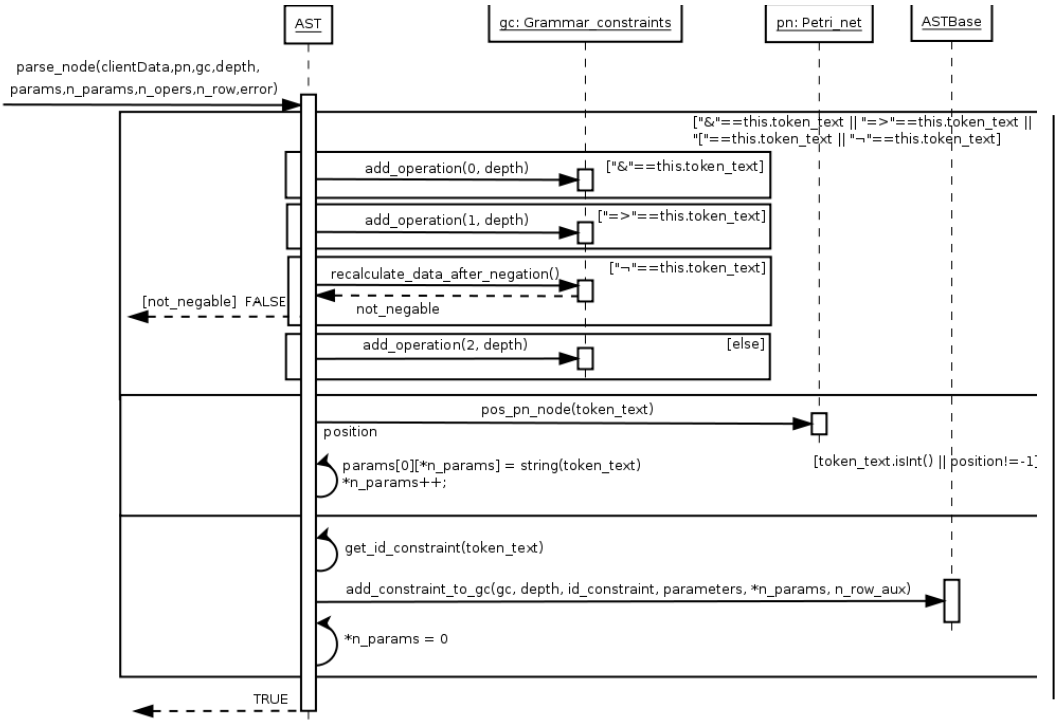


Figura 43: Diagrama de la funció que llegeix un token de l'arbre i valida la seva correctesa lèxica i sintàctica.
En el cas que sigui correcte també afegeix a l'estructura de propietats el token.

La funció **add_constraint_to_gc()** afegeix una nova entrada a l'estructura de dades que conté les propietats amb les dades que es passen per paràmetre.

4.3.2.2.2 Diagrama del cas d'ús "Negar propietat"

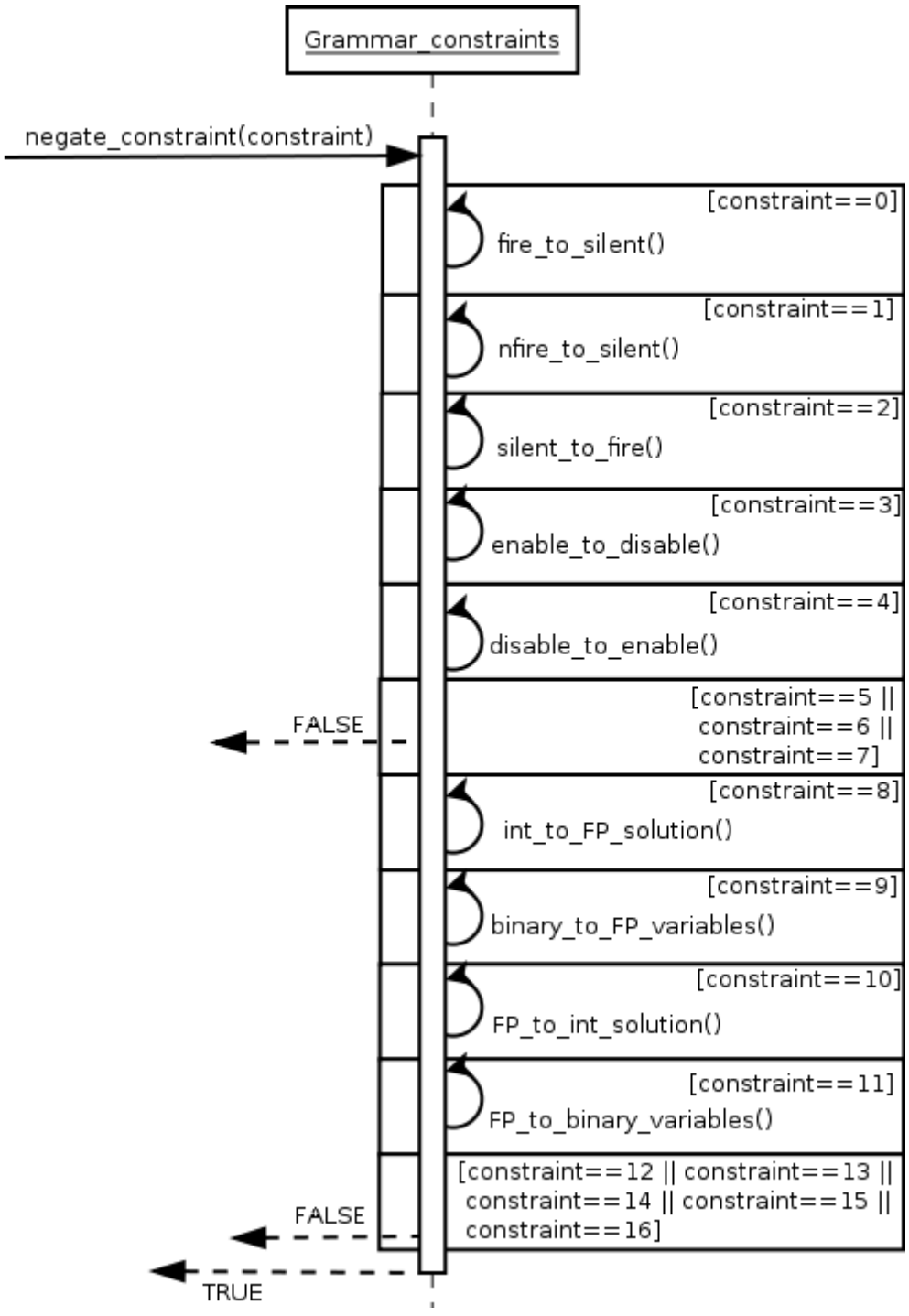


Figura 44: Cada sub-funció modifica els identificadors i els paràmetres de les funcions convenientment per tal de codificar la negada de la restricció en qüestió. Es retorna fals quan no es permet negar una restricció.

4.3.2.2.3 Diagrama del cas d'ús "Solucionar problema"

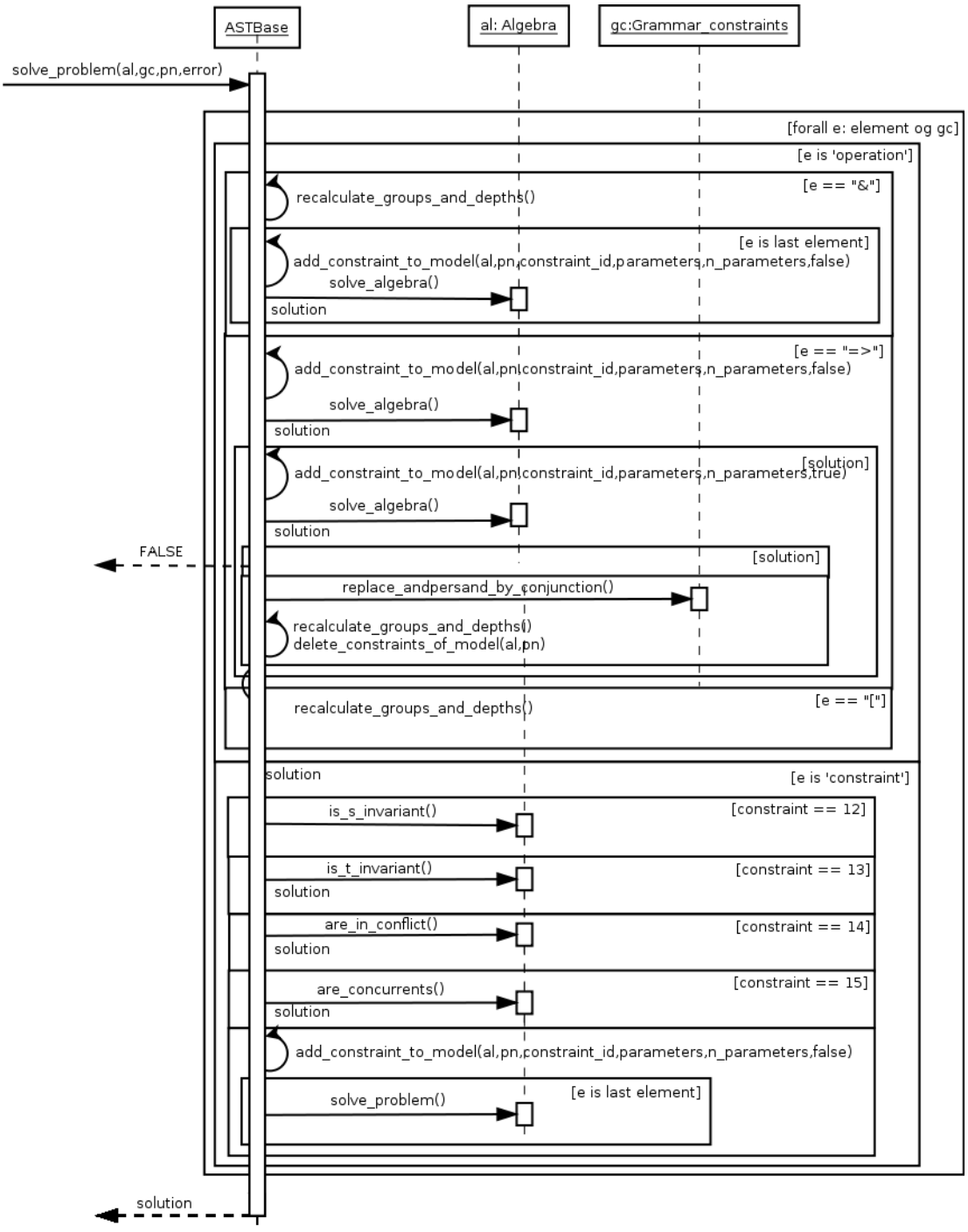


Figura 45: La funció *recalculate_groups_and_depths()* va simplificant el problema a mida que el va llegint ajuntant el màxim nombre de restriccions que puguin ser executades a la vegada sense trencar l'ordre de prioritats establert.

4.4 Patrons de disseny

Un **patró de disseny** és un esquema de com solucionar un problema de disseny que pot ser utilitzat en múltiples situacions.

Existeixen diferents tipus de patrons, que es poden organitzar en tres grups:

- Patrons creacionals: afecten a la inicialització i la configuració dels objectes.
- Patrons estructurals: separen la interfície de la implementació. Afecten a l'agrupació de les classes i els objectes.
- Patrons de comportament: afecten a la descripció de la comunicació entre els objectes.
- GRASP, General Responsability Assignement Software Patterns (Patrons Generals de Software per a l'Assignació de Patrons): conjunt de bones pràctiques que es recomana aplicar durant el disseny de software i que tracten sobre l'assignació de responsabilitats entre les classes (alta cohesió, baix acoblament, creador, etc.).

Per al desenvolupament del sistema s'ha utilitzat tècniques GRASP. A continuació s'expressa la base teòrica de les mateixes i quina ha estat la seva aplicació al codi.

4.4.1 Patró creador i patró expert

L'assignació de **responsabilitats** a objectes consisteix a determinar quines són les obligacions concretes dels objectes del diagrama de classes a l'hora de donar resposta als esdeveniments externs.

El **patró creador** s'aplica per a determinar qui ha de tenir la responsabilitat de crear una nova instància d'una classe. La seva aplicació pretén mantenir un baix acoblament.

Aquest patró s'ha aplicat en la creació de l'estructura que conté una XdP. La classe Petri_net té la responsabilitat de crear objectes de tipus Place, Transition i Arc.

Figura 46: Patró Creador aplicat al sistema

El **patró expert** ajuda a decidir a quina classe se li ha d'assignar una responsabilitat concreta. Aquest patró beneficia en termes d'acoblament, fent que resti baix degut a que es manté l'encapsulament, i de cohesió, que resta alta gràcies a la conducta distribuïda entre les classes que tenen la informació.

Aquest patró s'ha aplicat en tota l'aplicació, dividint el sistema en les mínimes unitats possibles i fent que les comunicacions mantinguin el mínim acoblament possible.

4.4.2 Patró controlador façana

El sistema rep esdeveniments que han de ser interceptats i tractats convenientment. El patró controlador façana consisteix a crear una classe, el **controlador**, que s'encarrega de rebre tots els esdeveniments que li arriben al sistema i de delegar en un o més objectes del sistema el tractament d'aquest esdeveniment.

A continuació es mostra l'esquema que representa l'aspecte estàtic del patró, on s'observa la classe *Verification_iterface*, una classe singletó que conté tantes operacions com esdeveniments ha de capturar el sistema, i la resta de classes, que tracten aquests esdeveniments i es comuniquen entre elles.

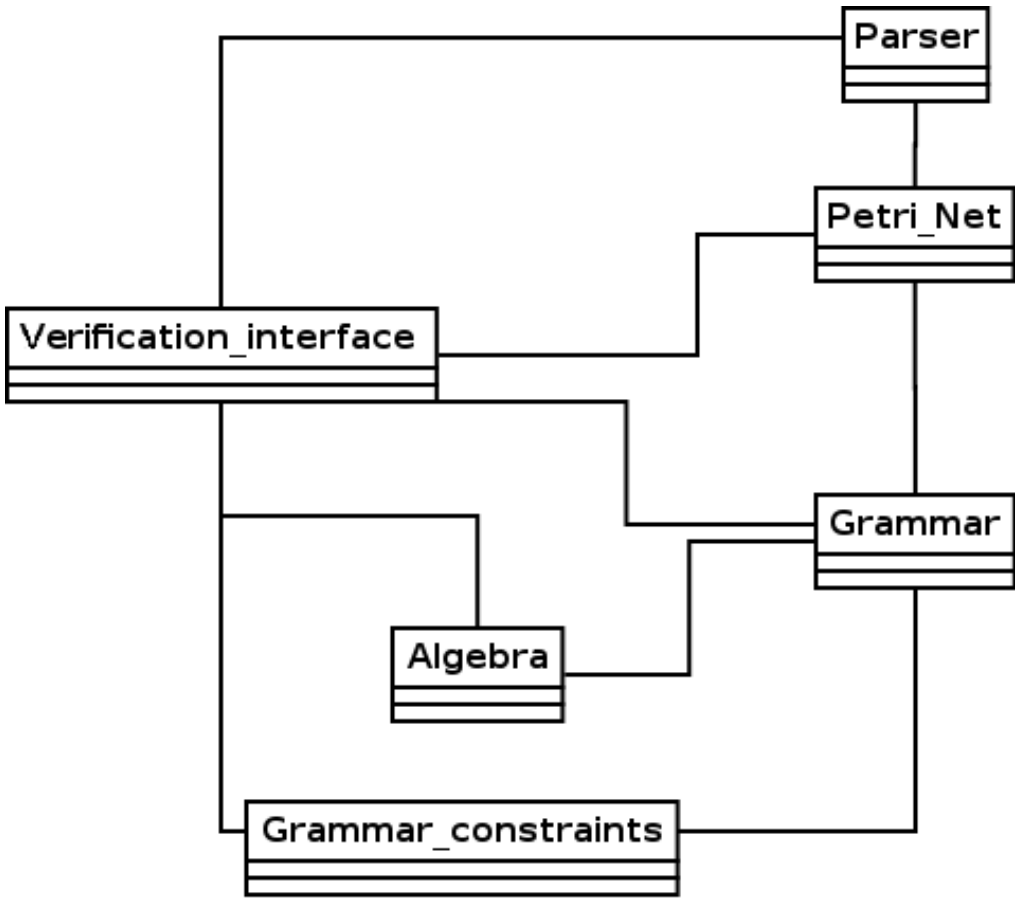


Figura 47: Patró Façana aplicat al sistema

En el sistema, la classe Verification_interface captura tots els esdeveniments que ocorren i els reparteix, segons el seu àmbit, entre les diferents classes de la llibreria (Petri_net, Parser, Algebra, Grammar, Grammar_constraints).

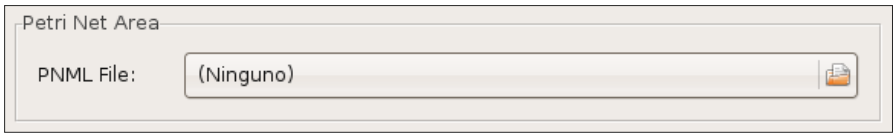
4.5 Interfície: mapes navegacionals

La interfície s'ha dissenyat tal i com es mostra a l'especificació.

Per a verificar un conjunt de propietats sobre una XdP cal seguir els següents passos:

1. Escollir la XdP amb la que es vol treballar a l'àrea A:
    a. Seleccionar el fitxer amb la XdP des del navegador de fitxers, component 3:

1. Botó per accedir al navegador de fitxers:



2. Navegador de fitxers:

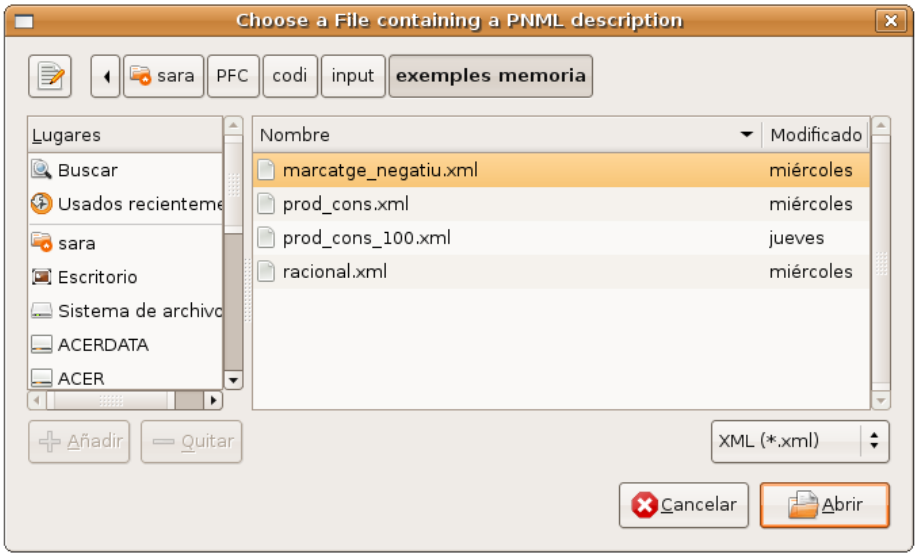


Figura 48: Selecció d'una fitxer amb una XdP a la interfície

b. Si la XdP és vàlida, es carrega una imatge creada de la XdP a l'àrea 2:

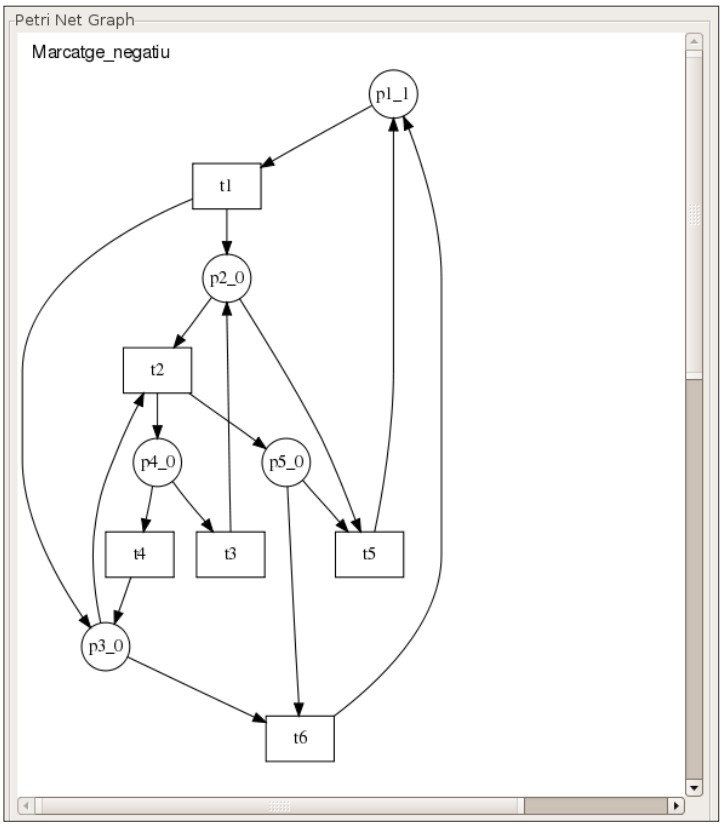


Figura 49: Imatge creada per l'eina a partir de l'anàlisi d'una XdP

c. Si la XdP no és vàlida, la interfície mostra el missatge corresponent:

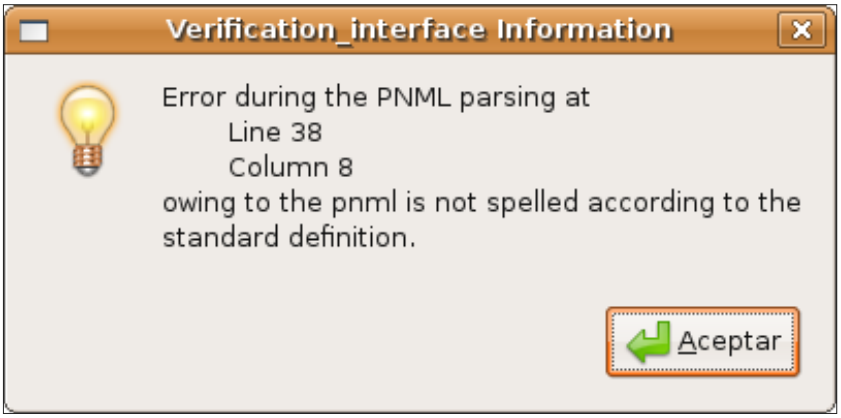


Figura 50: Missatge amb l'error trobat durant l'anàlisi de la xarxa, per exemple, un error a la línia 38 degut a que l'xml, tot i ser un xml ben format, no segueix l'estructura descrita per aquesta aplicació.

2. Introduir les propietats que es vol verificar a l'àrea 3 i verificar:
   a. Cal seleccionar el nombre de traces amb les que volem treballar amb el component 4:

Figura 51: Quadre per seleccionar el nombre de traces dins de l'àrea de les Propietats de la interfície

   b. Introduir les propietats que es vol verificar al component 5:

Figura 52: Quadre per introduir les Propietats dins de l'àrea de les Propietats de la interfície

   c. Verificar les propietats amb el component 8:

Figura 53: Botó per validar les Propietats dins de l'àrea de les Propietats de la interfície

d. Com a resultat de la operació, la interfície obrirà una nova finestra amb el missatge corresponent:

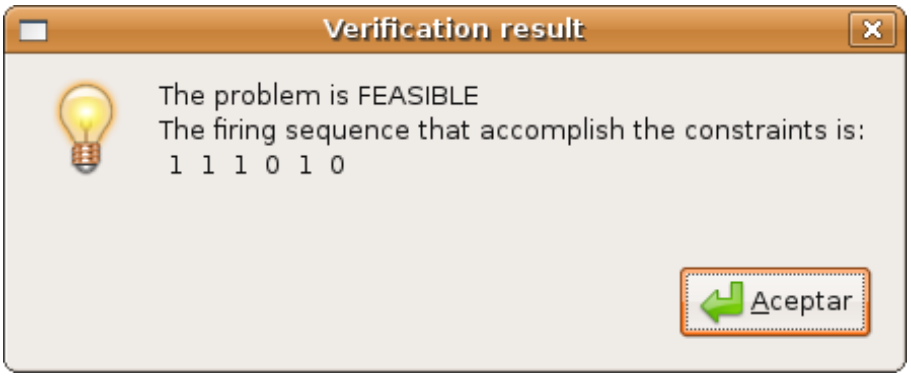


Figura 54: Missatge que retorna el conjunt de propietats definit anteriorment sobre la XdP escollida. El missatge indica que les propietats es compleixen i dóna un vector de dispars que dóna una solució que les compleix.

3. Veure el fitxer de registres, que conté els registres que han anat deixant les diferents operacions realitzades pel sistema:

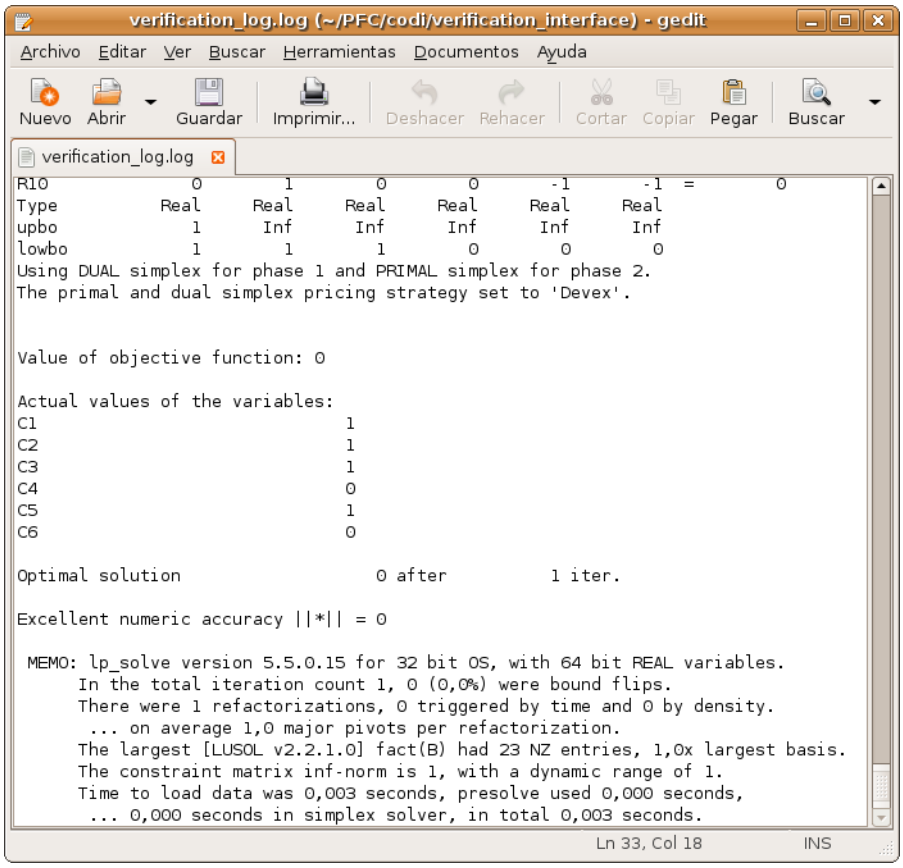


Figura 55: Part final del Fitxer de Registres amb les operacions realitzades pel sistema a l'anterior exemple.

5 Implementació

5.1 Tecnologies usades durant el desenvolupament

Durant el desenvolupament s'ha seguit la màxima d'utilitzar únicament programari lliure, així que les decisions sobre les tecnologies estan influenciades per aquest fet.

Als següents punts s'exposa els programes utilitats i el seu ús.

5.1.1 Eclipse

Eclipse és una plataforma de desenvolupament multi-llenguatge de codi lliure. Tot i que es va desenvolupar per la creació d'aplicacions Java, gràcies a les extensions proporcionades per una gran quantitat de plugins existents ha evolucionat pe acabar sent un *framework*[4] genèric adaptable al desenvolupament de qualsevol producte, no només aplicacions informàtiques, sinó també creació de documents, disseny d'aplicacions, etc.

Les avantatges principals d'aquesta plataforma són la seva naturalesa *open source*, la independència respecte del sistema operatiu, l'escalabilitat i extensibilitat, el fet de tenir un editor intel·ligent molt avançat i la disponibilitat d'actualitzacions constantment.

Els inconvenients principals són l'elevat ús de memòria que fa i certes dificultats que poden aparèixer en integrar diferents versions de plugins i eclipse.

S'ha escollit treballar amb un IDE (Integrated Development Environment) com Eclipse pel fet d'incorporar editor intel·ligent, compilador, debugger i eines de compilació automàtiques en un mateix entorn. Es va fer proves amb editors, com VIM (amb alguns plugins propis) i JEdit, i es va crear un makefile per compilar el programa des de la consola d'administració, però aquests processos incrementaven el temps perdut en tasques que no eren purament de desenvolupament.

[4] S'entén com **framework** una estructura conceptual i tecnológica de suport definit sobre la qual es pot organitzar i desenvolupar un altre projecte.

5.1.2 Pipe2, Platform Independent PetriNet Editor 2

Pipe2 és una eina de naturalesa *open source* i independent de la plataforma per dibuixar i analitzar XdP.

S'ha utilitzat aquest eina per dibuixar les XdP durant el disseny de l'aplicació davant d'altres opcions com SmartDraw (eina de pagament) o Tapaal (eina lliure, però específica per XdP amb temps) degut a que s'ajusta perfectament a la necessitat de poder dibuixar i exportar fàcilment les XdP. A més, ha permès fer anàlisis bàsics sobre la XdP que han servit per contrastar certes solucions obtingudes amb el sistema desenvolupat durant l'anàlisi.

5.1.3 Dia

Dia és un programa per la creació de diagrames per sistemes Unix i Windows, amb llicència GPL[5] i inspirat en MSVisio. Entre d'altres, permet dibuixar diagrames UML, i és pel que s'ha utilitzat en aquest projecte.

A continuació es mostra una petita comparativa sobre els aspectes que ens interessa dels programes testejats per al disseny dels diagrames d'aquest document:

	Dia	**MSVisio**	**ArgoUML**
Llicència	Lliure	MicroSoft	Lliure
Plataforma	Linux / Windows	Windows	Java
Resultats exportant imatges	Bons	Molt bons	Regulars
Fàcil d'instal·lar	Sí	Depèn de l'entorn	Sí

S'ha escollit l'eina Dia davant les altres per que és open source, multi plataforma i, dins d'aquest paràmetres, és la que millor resultats en qualitat d'imatge ha donat en l'exportació del diagrames.

[5] **GPL, General Public License**

Llicència creada per la Fundació de Software Lliure (Free Software Foundation) amb la finalitat de protegir la distribució, la modificació i l'ús de software impedint qualsevol intent d'apropiació que restringeixi les llibertats dels usuaris.

5.2 Llenguatges de programació

5.2.1 C++

El llenguatge C++, nascut amb la finalitat d'estendre el llenguatge C, és un llenguatge procedural (orientat a algoritmes) i orientat a objectes.

Es va barallar la opció de realitzar el projecte amb altres llenguatges com C o Java, però es van desestimar. Per als dos casos es va trobar mancances en el llenguatge que afectaven el tractament de les dades que ha de representar el sistema. A continuació s'exposa els inconvenients trobats per cadascun dels llenguatges :

- **C** no té suport per a la orientació a objectes, no permet herència de classes i la gestió de la memòria l'ha de fer explícitament el programador.
- **Java** no permet un accés a memòria a baix nivell (ús de punters), el pas de paràmetres per referència ni l'herència múltiple.

A més, a favor de C++ dir que és compatible amb el codi C, pot ser compilat per a múltiples sistemes operatius sense necessitat de modificar el codi, consta d'una llibreria estàndard que dóna solució a totes les necessitats del projecte i permet manejar la memòria tant de forma explícita com de forma implícita per molts casos.

5.2.2 XML

La codificació d'entrada per les XdP es fa amb **PNML**, una especificació de XdP basada en **XML** (veure definició més detallada a dins de l'apartat 3.1.1 d'aquest document). S'ha utilitzat la metodologia DOM[6] per accedir a la informació del fitxer XML; s'ha

escollit aquesta, i no SAX[7] (Simple Api for XML), per que construeix un arbre a memòria que permet un accés més senzill i no seqüencial les dades.

Per verificar que la XdP d'entrada és un fitxer que conté una estructura XML vàlida s'ha definit un **XSDSchema** que defineix una subclasse de XdP que identifica els elements bàsics d'una xarxa de petri (identificador, llocs, transicions, arcs i tots els seus atributs) i certs aspectes sobre la seva representació gràfica que poden ser utilitats per algunes aplicacions per dibuixar les xarxes.

---

[6] **DOM, Document Object Model**

Metodología de programació d'aplicacions que proporciona un conjunt estàndard d'objectes per representar documents de tipus HTML i XML, un model estàndard sobre com es poden combinar aquests objectes i una interfície per accedir-los i manipular-los.

[7] **SAX, Simple API for XML**

És una interfície simple per aplicacions XML inicialment adaptada per Java que s'utilitza especialment en situacions en què els arxius XML ja estan en una forma estructuralment similar a la que volem obtenir.

5.3 Llibreries

5.3.1 Xerces-C++ 2.8

Xerces-C++ és un analitzador sintàctic escrit en C++ per a validar documents XML. La llibreria utilitzada permet analitzar, generar, manipular i validar documents usant diferents metodologies (DOM, SAX i SAX2).

El sistema utilitza aquesta llibreria per analitzar, manipular i validar els fitxers d'entrada que contenen un document amb format XML amb la XdP. Per a realitzar aquestes tasques es va valorar utilitzar una altra llibreria, **libxml2**. Totes dues són projectes àmpliament desenvolupats i testejats i proporcionen les funcionalitats requerides, però es va optar per Xerces-C++ per que està millor documentat.

L'esquema, XSDSchema, que defineix una XdP vàlida per al sistema és el següent:

```
<?xml version="1.0" encoding="utf-16"?>

<xsd:schema       attributeFormDefault="unqualified"       elementFormDefault="qualified"
version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="pnml" type="pnmlType" />

  <xsd:complexType name="pnmlType">
    <xsd:all>
      <xsd:element name="net" type="netType" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="netType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="name" type="nameType" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="dscriptn" type="dscriptnType" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="graphics" type="graphicsType" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="place" type="placeType"/>
     <xsd:element name="transition" type="transitionType"/>
     <xsd:element name="arc" type="arcType"/>
    </xsd:choice>
    <xsd:attribute name="type" type="xsd:string" use="required" />
    <xsd:attribute name="id" type="xsd:ID" use="required" />
  </xsd:complexType>

  <xsd:complexType name="nameType">
    <xsd:all>
      <xsd:element name="text" type="xsd:string" />
      <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="descriptionType">
    <xsd:all>
      <xsd:element name="text" type="xsd:string" />
      <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="placeType">
    <xsd:all>
      <xsd:element name="name" type="nameType" minOccurs="0" />
      <xsd:element name="initialMarking" type="initialMarkingType" minOccurs="0" />
      <xsd:element name="type" type="typeType" minOccurs="0" />
      <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
    </xsd:all>
    <xsd:attribute name="id" type="xsd:ID" use="required" />
  </xsd:complexType>
```

```
  <xsd:complexType name="initialMarkingType">
    <xsd:all>
     <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
      <xsd:element name="text" type="xsd:string" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="typeType">
    <xsd:all>
      <xsd:element name="text" type="xsd:string" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="transitionType">
    <xsd:all>
      <xsd:element name="name" type="nameType" minOccurs="0" />
      <xsd:element name="type" type="typeType" minOccurs="0" />
      <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
    </xsd:all>
    <xsd:attribute name="id" type="xsd:ID" use="required" />
  </xsd:complexType>

  <xsd:complexType name="graphicsType">
    <xsd:all>
      <xsd:element name="offset" type="offsetType" minOccurs="0" />
      <xsd:element name="dimension" type="dimensionType" minOccurs="0" />
      <xsd:element name="position" type="positionType" minOccurs="0" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="inscriptionType">
    <xsd:all>
      <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
      <xsd:element name="text" type="xsd:string" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="arcType">
    <xsd:all>
      <xsd:element name="graphics" type="graphicsType" minOccurs="0" />
      <xsd:element name="inscription" type="inscriptionType" minOccurs="0" />
    </xsd:all>
    <xsd:attribute name="id" type="xsd:ID" use="required" />
    <xsd:attribute name="source" type="xsd:IDREF" use="required" />
    <xsd:attribute name="target" type="xsd:IDREF" use="required" />
  </xsd:complexType>

  <xsd:complexType name="dimensionType">
    <xsd:attribute name="page" type="xsd:int" minOccurs="0" />
    <xsd:attribute name="x" type="xsd:integer" />
    <xsd:attribute name="y" type="xsd:integer" />
  </xsd:complexType>

  <xsd:complexType name="positionType">
    <xsd:attribute name="page" type="xsd:int" minOccurs="0" />
    <xsd:attribute name="x" type="xsd:integer" />
    <xsd:attribute name="y" type="xsd:integer" />
  </xsd:complexType>

  <xsd:complexType name="offsetType">
    <xsd:attribute name="page" type="xsd:int" minOccurs="0" />
    <xsd:attribute name="x" type="xsd:integer" />
    <xsd:attribute name="y" type="xsd:integer" />
  </xsd:complexType>

</xsd:schema>
```

5.3.2 Log4cpp

Log4cpp és una llibreria de C++ destinada a facilitar la tasca de guardar els registre (logs) d'una eina ja sigui a arxius, sortida estàndard o altres destinacions.

S'ha desenvolupat sobre el model de la llibreria Log4j, dissenyada per aplicacions Java.

Permet classificat els registres segons el seu nivell (informació, avís, error,...) i es poden gestionar automàticament a partir d'una única crida.

S'ha usat aquesta llibreria per que era ja coneguda i s'havia contrastat la seva utilitat.

5.3.3 Lp_solve 5.5

Lp_solve és un solucionador **MILP**, Mixed Integer Linear Programming (PLE), basat en el mètode **Simplex** pels Reals i el mètode de **Ramificació i Poda** (Branch-and-bound) pels Enters. Es pot trobar més informació sobre aquests mètodes a l'apartat 2.2 d'aquest document.

Proporciona diferents maneres d'introduir dades a la llibreria: una API (Application Programming Interface), un IDE o fitxers d'entrada en un format específic (MPS i LP)[8].

El model no té límits de mida: com més gran sigui el model, més fàcil és trobar una solució.

Aquesta llibreria s'utilitza per la verificació de propietats sobre les XdP, prèvia transformació de les propietats en un conjunt de restriccions de programació lineal que són introduïdes en el solucionador.

[8] **MPS** és un format orientat a columnes (oposadament al model d'equacions) on totes les dades venen etiquetades per un nom.

LP és el format natiu de lpsolves per llegir i escriure models, que es basa en expressions algebraiques.

A continuació es mostra un exemple extret de la web de lp_solve (7) molt simple problema de programació lineal codificat amb aquests dos formats:

NAME TEST ROWS N COST L LIM1 COLUMNS XONE COST 1 LIM1 1 YTWO COST 4 LIM1 1 RHS RHS1 LIM1 5 BOUNDS UP BND1 XONE 4 LO BND1 YTWO -1	minimitzar: XONE + 4*YTWO XONE + YTWO <= 5 XONE >= 4 YTWO >= -1
Problema de programació lineal	
Format MPS	Format LP

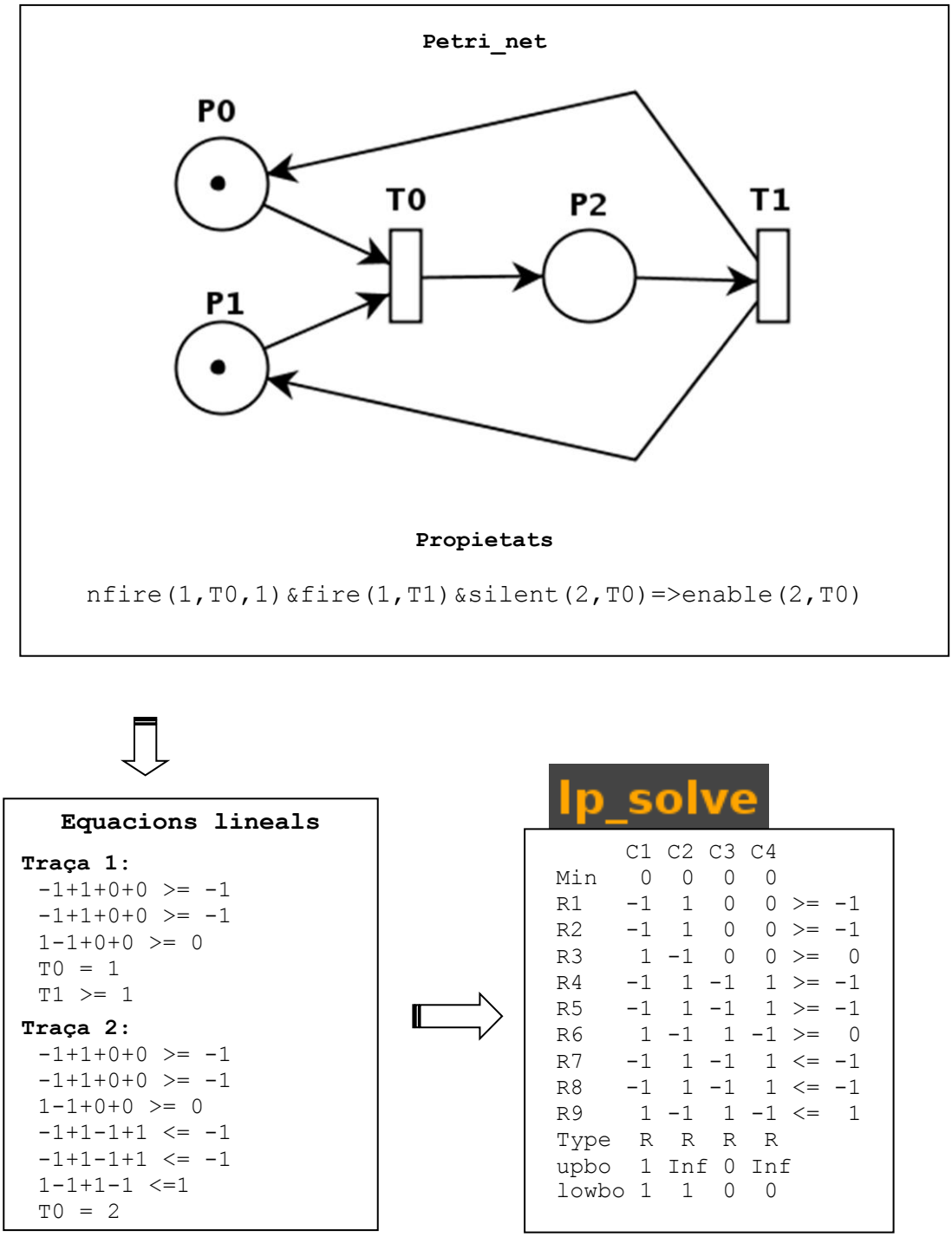


Figura 56: Exemple de transformació d'un conjunt de propietats sobre una XdP en un conjunt de inequacions de PL i, a continuació, en un problema generat amb lp-solve.

Existeixen molts altres solucionadors i llenguatges d'"scripting" per a problemes de programació lineal, alguns de caire comercial com per exemple CPLEX i Xpress, i altres de lliure distribució, com GLPK (suporta diferents format d'entrada). Es va escollir Lp_solve a proposició del tutor de projecte per ser gratuït i per estar àmpliament testejat i oferir el millor rendiment.

5.3.4 PCCTS 1.33

PCCTS és una eina que permet crear compiladors a partir d'un únic fitxer on s'especifica:

- Els tokens a reconèixer (anàlisi lèxica).
- La gramàtica (anàlisi sintàctica).
- Les accions a aplicar (programa principal amb les crides als altres components).

Treballa amb llenguatges LL(k)[9] amb k>=1 mitjançant un mètode d'anàlisi sintàctica descendent.

PCCTS es composa de tres eines:

- **DLG**: generador d'analitzadors lèxics. Té com a entrada una descripció dels tokens vàlids i crea l'AFD[10] corresponent.
- **ANTLR, ANother Tool for Language Recognition**: generador d'analitzadors sintàctics. Té com a entrada una descripció de la gramàtica usant la notació EBNF[11] i construeix l'analitzador descendent associant una funció a cadascuna de les produccions de la gramàtica.
- **SOURCERER**: generador de traductors de codi font a codi font en un altre llenguatge.

La llibreria PCCTS s'utilitza en aquest sistema per l'anàlisi de la cadena de text que introdueix l'usuari i que representa les propietats que el sistema haurà de verificar.

Per poder usar-la ha calgut fer una feina prèvia per tal de generar els fitxers en llenguatge C++ que ens permetin validar les propietats. De les tres eines que disposa PCCTS, n'hem hagut d'utilitzar dues: ANTLR i DLG. A continuació es fa una descripció exhaustiva dels passos que s'ha seguit i es mostra un esquema general del procés:

1. Crear un fitxer amb extensió .g que contingui la descripció lèxica i sintàctica de la gramàtica. En aquest fitxer es defineix, tal i com s'ha dit anteriorment:
   - Com es farà l'anàlisi: a partir de la funció *main*.
   - Quins són els tokens que s'acceptaran: *#token nom_token "valor_token"*.
   - Quina és la gramàtica: *class nom { producció1; ... producciónN; }*.

---

[9] **LL(k)**, llenguatge que pot ser analitzat per un analitzador LL.

Un analitzador LL és un analitzador descendent que tracta les entrades d'esquerra a dreta i les construccions de les derivacions per l'esquerra.

[10] **AFD, Autòmat Finit Deterministic**

Màquina d'estats finita on, per a cada parell d'estat i símbol d'entrada hi ha una i només una transició a un estat futur.

[11] **EBNF, Extended Backus–Naur Form**

Metasintaxi utilitzada per expresar gramàtiques lliures de context, és a dir, una manera formal de descriure llengutges formals.

Una especificació de BNF és un sistema de regles de derivació, escrit de la forma:

```
<simbol> = <expressió amb símbols>
```

```
<<
typedef ANTLRCommonToken ANTLRToken;
#include "DLGLexer.h"
#include "PBlackBox.h"

class AST : public ASTBase {
public:
  ANTLRTokenPtr token;
  AST(ANTLRTokenPtr t) { token = t; }
  void preorder_action(void * clientData) {
    char *s = token->getText();
    printf("\n %s", s);
  }
};

int main(int argc, char* argv[]) {
  DLGStringInput in(argv[1]);   /* create an input stream for DLG */
  DLGLexer scan(&in);           /* create scanner reading from stdin */
  ANTLRTokenBuffer pipe(&scan); /* make pipe between lexer & parser */
  ANTLRTokenPtr aToken = new ANTLRToken;
  scan.setToken(mytoken(aToken));
  Expression parser(&pipe);     /* create parser of type Expression
                                hooked to scanner */
  parser.init();                /* init the parser */

  ASTBase *root = NULL;
  parser.expression(&root);     /* start parsing at rule 'expression' */
  root->inorder_prety_print();
  return 0;
}
>>

#token       "[\ \t\n]*" <<skip();>>
#token VAL   "[a-zA-Z0-9]+"
#token NOT   "¬"
#token IMPL  "=>"
#token AND   "&"
#token OPPAR "\("
#token CLPAR "\)"
#token OPCOR "\["
#token CLCOR "\]"
#token COMMA ","
#token Eof   "@"


class Expression {
  expression:expressionrght (AND^ expressionrght | IMPL^ constraint)*;
  expressionrght: (OPCOR^ expression CLCOR!) | ((NOT^ | ) constraint);
  constraint:VAL^ OPPAR! ( params | ) CLPAR!;
  params: VAL (COMMA! VAL)*;
}
```

Figura 57: Fitxer .g amb la descripció lèxica i sintàctica de la gramàtica

2. Cridar la comanda **antlr** per generar els fitxers encarregats de validar la sintaxis de la gramàtica (Grammar.g -> {Grammar.cpp, Expression.h/.cpp}), així com els fitxers que permetran generar el codi per reconèixer el lèxic (Grammar.g -> {parser.dlg, tokens.h}).
3. Cridar la comanda **dlg** per generar els fitxers que reconeixeran el lèxic de la gramàtica (parser.dlg -> {DLGLexer.h/.cpp}).

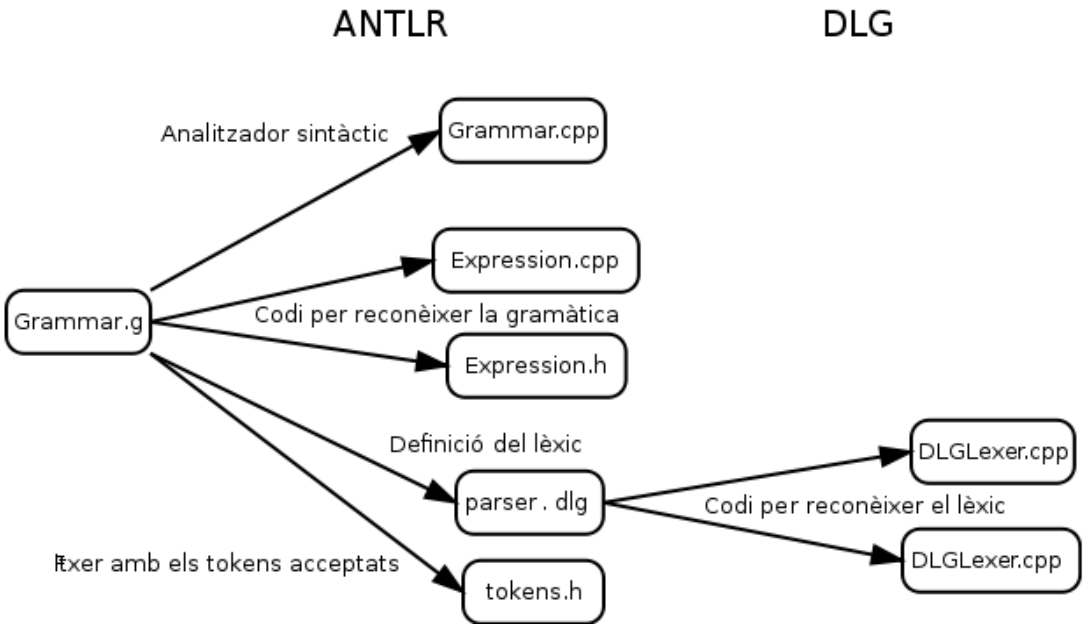


Figura 58: Procés de creació dels fitxers C++ de la gramàtica amb ANTLR i DLG

5.3.5 GraphViz

GraphViz és un programa de disseny que, a partir de la descripció d'un graf en format de text pla, permet generar diagrames en diferents formats.

Implementa diferents tipus d'algoritmes de disseny de grafs, dels quals s'ha utilitzat **dot**, un algoritme per dibuixar grafs dirigits en forma de jerarquies que té com a objectiu dibuixar els arcs del graf en la mateixa direcció (de dalt a baix o de dreta a esquerra), tractant d'evitar encreuaments i de reduir la longitud dels arcs.

Donat que l'usuari introdueix les XdP en format PNML, abans de cridar la llibreria per generar la imatge cal crear un fitxer .dot amb la informació necessària per descriure les característiques de la XdP. Aquest fitxer té l'estructura que es mostra a continuació:

```
digraph nom_XdP {
  label= nom_XdP; labelloc=t; labeljust=l;
  dpi=75; fontname=Helvetica; ratio=fill;

  p1 [shape=circle, label=a11, font=Helvetica, margin=0,0];
  ...
  pN [shape=circle, label=a11, font=Helvetica, margin=0,0];

  t1 [shape=box, font=Helvetica, margin=0,0];
  ...
  tM [shape=box, font=Helvetica, margin=0,0];

  p1 -> t1 [len=.5];
  ...
  pN -> tN [len=.5];
}
```

Figura 59: Esquema del fitxer .dot generat a partir d'una XdP

5.3.6 wxWidgets

wxWidgets és una API per escriure applicacions amb GUI multiplataforma. És funcional i està molt ben documentada. Ha permés realitzar una interfície de forma ràpida i completa, permetent, d'una forma visual, que l'usuari pugui realitzar totes les tasques implementades.

Es va fer una prova amb gtk+, però la prova amb wxWidgets va ser molt més ràpida i els resultat final millor.

5.4 Requeriments

En quan a les necessitats de l'eina per a funcionar en una màquina, les podem dividir en segons si sñón de Hardware o de Software.

- **Requeriments Hardware**

  PN_Verification és una eina que no requereix un ordinador amb molta potència de càlcul, un Pentium III 1800MHz o similar és suficient. La memòria és més important, depenent del volum de dades que estem tractant, però s'ha testejat amb 1Gb i per volums de dades mitjanament grans (una xarxa amb 1000 nodes i una verificació de propietats amb 10 traces) es garanteix un correcte funcionament.

- **Requeriments Software**

  PN-Verification s'ha testejat a sistemes Ubuntu, però està pensat per qualsevol distribució de Linux.

  La majoria de les llibreries utilitzades pel sistema estan incloses al codi. Però cal que el sistema on es faci còrrer l'aplicació hi hagi instal·lat Graphviz i wxWidgets. Aquestes llibreries es poden trobar a http://www.graphviz.org/Download_linux_ubuntu.php i http://www.wxwidgets.org/downloads/, respectivament.

6 Proves

A continuació es mostra una sèries de proves fetes sobre el sistema que es consideren d'interès per veure tant el potencial com les limitacions del mateix. Es posa de manifest la capacitat de càlcul de la eina desenvolupada i es verifica els resultats.

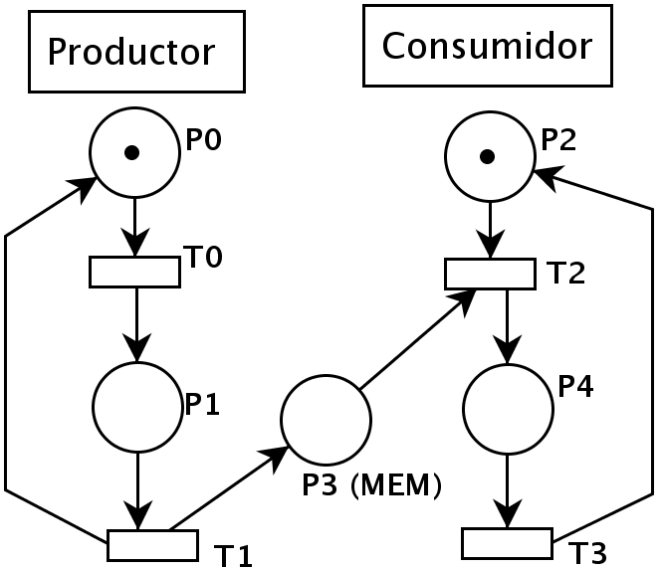
6.1 Productor – Consumidor

El problema anomenat "Productor - Consumidor" és un exemple típic que il·lustra la necessitat de sincronitzar sistemes on diferents processos comparteixen un recurs.

Farem dos exemples d'aquest sistema, un de petit i un de gran, per posar de manifest diferents aspectes de l'eina com són les possibilitats i la potència de càlcul.

6.1.1 Model petit

Consisteix en dos processos, el productor, que produeix informació i la guarda a memòria, i el consumidor, que requereix d'aquesta informació per continuar la seva execució.



$$A = \begin{pmatrix} -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

$$M_0 = (1 \quad 0 \quad 0 \quad 1 \quad 0)$$

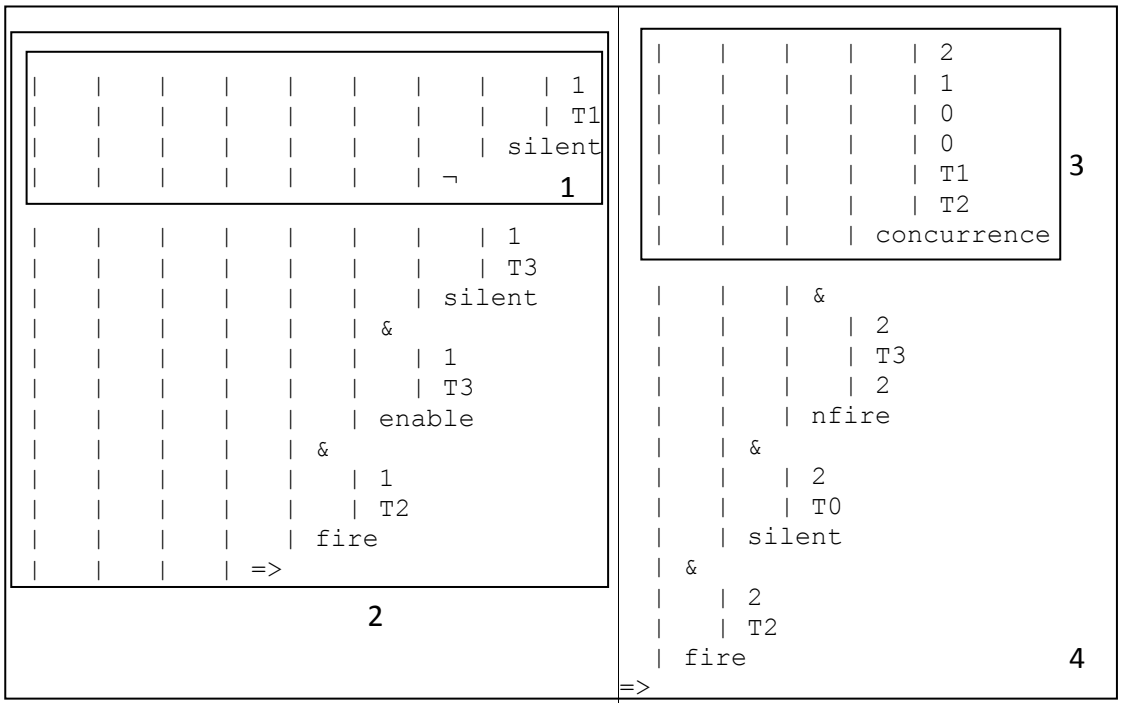
Figura 60: XdP que representa el procés d'un sistema Productor-Consumidor amb la seva matriu d'incidència i el marcatge inicial

Es vol verificar que existeix solució per al sistema donades unes condicions de producció i consum per a dues traces, i que existeix concurrència entre dues transicions en un moment donat del procés.

La propietat a verificar és la següent:

fire(1,T1)&silent(1,T3)&enable(1,T3)=>fire(1,T2)&concurrence(2,1,0,0,T1,T2)& nfire(2,T3,2)&silent(2,T0)=>fire(2,T2).

El sistema llegeix les propietats i genera el següent arbre llegit en inordre:



Ordre de resolució de les propietats:

1. Negació d'una propietat, que es abans d'introduir-la al model: la propietat silent passar a ser fire.
2. Primera implicació on la part esquerra són totes les propietats que estan abans amb la prioritat menor.
3. Propietat de concurrència, que es soluciona fora del model comú.
4. Darrera implicació, amb tots els elements anteriors com part esquerra de la operació, a excepció de la concurrència.

Figura 61: Arbre generat amb la gramàtica per la propietat fire(1,T1)&silent(1,T3)&enable(1,T3)=>fire(1,T2)&concurrence(2,1,0,0,T1,T2)& nfire(2,T3,2)&silent(2,T0)=>fire(2,T2).

El model de PL es soluciona en tres passos, mostrats a les següents figures:

```
        C1    C2    C3    C4    C5    C6    C7    C8
Min      0     0     0     0     0     0     0     0
R1      -1     1     0     0     0     0     0     0 >=    -1
R2       1    -1     0     0     0     0     0     0 >=     0
R3       0     0    -1     1     0     0     0     0 >=    -1
R4       0     1    -1     0     0     0     0     0 >=     0
R5       0     0     1    -1     0     0     0     0 >=     0
R6      -1     1     0     0    -1     1     0     0 >=    -1
R7       1    -1     0     0     1    -1     0     0 >=     0
R8       0     0    -1     1     0     0    -1     1 >=    -1
R9       0     1    -1     0     0     1    -1     0 >=     0
R10      0     0     1    -1     0     0     1    -1 >=     0
R11     -1     1     0     0     0     0     0     0 >=    -1
R12      1    -1     0     0     0     0     0     0 >=     0
R13      0     0    -1     1     0     0     0     0 >=    -1
R14      0     1    -1     0     0     0     0     0 >=     0
R15      0     0     1    -1     0     0     0     0 >=     1
Type     R     R     R     R     R     R     R     R
upbo   Inf   Inf     0     0   Inf   Inf   Inf   Inf
lowbo    0     1     0     0     0     0     0     0

Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:       15 constraints,       8 variables,           40 non-zeros.
Sets:                              0 GUB,                 0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

The model is INFEASIBLE
lp_solve unsuccessful after 1 iter and a last best value of 1e+30

MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 1, 0 (0,0%) were bound flips.
There were 0 refactorizations, 0 triggered by time and 0 by density.
... on average 1,0 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 16 NZ entries, 1,0x largest basis.
The constraint matrix inf-norm is 1, with a dynamic range of 1.
Time to load data was 0,002 seconds, presolve used 0,000 seconds,
... 0,000 seconds in simplex solver, in total 0,002 seconds.
```

Figura 62: Solució de lpsolve per a la propietat "¬silent(1,T1)&silent(1,T3)&enable(1,T3)&¬fire(1,T2)".

Com que no existeix cap solució que satisfaci totes les restriccions de l'esquerra de la implicació i la negada de la part dreta, llavors la implicació es compleix. Es substitueix l'operador de implicació pel de conjunció i es segueix.

Es calcula el marcatge després del vector de dispars indicat,

$$M_0 + Ax = \begin{pmatrix}1 & 0 & 0 & 1 & 0\end{pmatrix} + \begin{pmatrix}-1 & 0 & 1 & 0\\ 1 & 0 & -1 & 0\\ 0 & -1 & 1 & 0\\ 0 & -1 & 0 & 1\\ 0 & 1 & 0 & -1\end{pmatrix}\begin{pmatrix}2 & 1 & 0 & 0\end{pmatrix}$$
$$= \begin{pmatrix}0 & 1 & 1 & 1 & 0\end{pmatrix} = M_p$$

Es comprova que les transicions que es verifica estiguin activades per aquest marcatge:

- Pre-conjunt de T1: P1 està marcat.
- Pre-conjunt de T2: P2 i P3 estan marcats.

Es comprova la condició de concurrència: $|\cdot y| + |\cdot z| \leq \sum_{p \in \cdot y \cup \cdot z} M[p]$.

- $|\cdot y| = 1$,
- $|\cdot z| = 2$,
- $\sum_{p \in \cdot y \cup \cdot z} M[p] = 3$,
- $1 + 2 \geq 3$, es compleix la propietat de concurrencia.

Figura 63: Solució de la propietat “concurrence(2,1,0,0,T1,T2)”, independentment del model anterior, i es resol que sí existeix concurrència entre les dues transicions, T1 i T2.

```
       C1    C2    C3    C4    C5    C6    C7    C8
Min     0     0     0     0     0     0     0     0
R1     -1     1     0     0     0     0     0     0    >=    -1
R2      1    -1     0     0     0     0     0     0    >=     0
R3      0     0    -1     1     0     0     0     0    >=    -1
R4      0     1    -1     0     0     0     0     0    >=     0
R5      0     0     1    -1     0     0     0     0    >=     0
R6     -1     1     0     0    -1     1     0     0    >=    -1
R7      1    -1     0     0     1    -1     0     0    >=     0
R8      0     0    -1     1     0     0    -1     1    >=    -1
R9      0     1    -1     0     0     1    -1     0    >=     0
R10     0     0     1    -1     0     0     1    -1    >=     0
R11    -1     1     0     0     0     0     0     0    >=    -1
R12     1    -1     0     0     0     0     0     0    >=     0
R13     0     0    -1     1     0     0     0     0    >=    -1
R14     0     1    -1     0     0     0     0     0    >=     0
R15     0     0     1    -1     0     0     0     0    >=     1
Type    R     R     R     R     R     R     R     R
upbo   In    In   Inf     0     0   Inf   Inf     2
lowbo   0     1     1     0     0     0     0     2

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Value of objective function: 0
Actual values of the variables:
C1                            2
C2                            2
C3                            1
C4                            0
C5                            0
C6                            0
C7                            1
C8                            2
Optimal solution              0 after           3 iter.
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 3, 0 (0,0%) were bound flips.
      There were 1 refactorizations, 0 triggered by time and 0 by density.
       ... on average 3,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 21 NZ entries, 1,0x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,003 seconds, presolve used 0,000 seconds,
       ... 0,001 seconds in simplex solver, in total 0,004 seconds.
```

Figura 64: Solució de lpsolve pel problema complet independentment de la concurrència, ¬silent(1,T1)&silent(1,T3)&enable(1,T3)&fire(1,T2)& nfire(2,T3,2)&silent(2,T0)=>fire(2,T2).
Troba una solució, així que es verifica el conjunt de propietats.

6.1.2 Model gran

Per contrastar els resultats amb una XdP considerablement més gran, s'ha construït un sistema Productor – Consumidor amb 500 llocs i 500 transicions. A continuació es mostra l'esquema de la xarxa:

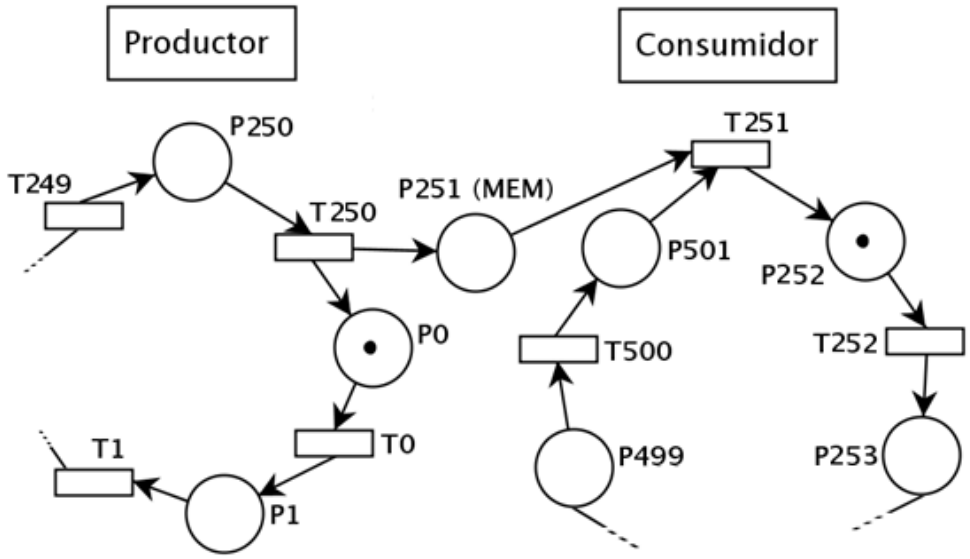


Figura 65: Esquema de la XdP que representa un sistema Productor-Consumido gran

Es vol contrastar el temps de càlcul i la capacitat del sistema de resoldre problemes més grans. Es comprova ara un conjunt de propietats sobre 10 traces. Per tal de donar unes dades intel·ligibles, s'ha posat una o dues restriccions per traça.

Les propietats a verificar son:

enable(1,T251)&silent(1,T252)&fire(2,T400)&ntokens(3,2,P251)&cyclicsolution(4)&
nfire(4,T35,2)&fire(5,T300)&disable(6,T15)&fire(6,T0)&silent(7,T300)&fire(7,T301)&
[cyclicsolution(8)&¬fire(8,T350)]=>fire(7,T350)
&enable(9,T25)&enable(9,T325)&nfire(10,T0,2)&fire(10,T499)

L'arbre, llegit en inordre, de les propietats és el que es mostra a continuació:

```
| | | | | | | | | | | | | | | | 1
| | | | | | | | | | | | | | | | T251
| | | | | | | | | | | | | | | enable
| | | | | | | | | | | | | | | | 1
| | | | | | | | | | | | | | | | T252
| | | | | | | | | | | | | | | silent
| | | | | | | | | | | | | | | &
| | | | | | | | | | | | | | | | 2
| | | | | | | | | | | | | | | | T400
| | | | | | | | | | | | | | | fire
| | | | | | | | | | | | | | | &
| | | | | | | | | | | | | | | | 3
| | | | | | | | | | | | | | | | 2
| | | | | | | | | | | | | | | | P251
| | | | | | | | | | | | | | | ntokens
```

```
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 4
| | | | | | | | | | | | | | cyclicsolution
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 4
| | | | | | | | | | | | | | T35
| | | | | | | | | | | | | | 2
| | | | | | | | | | | | | | nfire
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 5
| | | | | | | | | | | | | | T300
| | | | | | | | | | | | | | fire
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 6
| | | | | | | | | | | | | | T15
| | | | | | | | | | | | | | disable
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 6
| | | | | | | | | | | | | | T0
| | | | | | | | | | | | | | fire
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 7
| | | | | | | | | | | | | | T300
| | | | | | | | | | | | | | silent
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 7
| | | | | | | | | | | | | | T301
| | | | | | | | | | | | | | fire
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | | 8
| | | | | | | | | | | | | | cyclicsolution
| | | | | | | | | | | | | | | 8
| | | | | | | | | | | | | | | T350
| | | | | | | | | | | | | | fire
| | | | | | | | | | | | | | ¬
| | | | | | | | | | | | | | &
| | | | | | | | | | | | | | [
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 7
| | | | | | | | | | | | | | T350
| | | | | | | | | | | | | | fire
| | | | | | | | | | | | | | =>
| | | | | | | | | | | | | | 9
| | | | | | | | | | | | | | T25
| | | | | | | | | | | | | | enable
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 9
| | | | | | | | | | | | | | T325
| | | | | | | | | | | | | | enable
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 10
| | | | | | | | | | | | | | T0
| | | | | | | | | | | | | | 2
| | | | | | | | | | | | | nfire
| | | | | | | | | | | | | &
| | | | | | | | | | | | | | 10
| | | | | | | | | | | | | | T499
| | | | | | | | | | | | | fire
&
```

És impensable posar el model d'àlgebra lineal generat, ja que té **5000 variables**. El que sí que posarem, per donar una idea del de la càrrega de feina realitzada, són les dades que dóna el solucionador respecte de la cerca feta per l'espai de solucions.

Notar que en aquest cas, el solucionador ha hagut de fer **3867 iteracions** per l'espai de resultats, mentre que pel cas de l'exemple anterior, amb una xarxa amb quatre variables i dues traces, només amb 3 ja ha sigut capaç de donar solució. Això repercuteix en el temps de càlcul del model complet, que ha estat de 21,394 segons. Cal sumar el temps trigat a fer el càlcul de la implicació, que han estat uns 5 segons, el conjunt de les propietats s'ha verificat en **25 segons**.

```
Optimal solution              0 after       3867 iter.

Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 3867, 0 (0,0%) were bound flips.
      There were 16 refactorizations, 0 triggered by time and 0 by density.
       ... on average 241,7 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 80218 NZ entries, 1,0x
      largest basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 13,541 seconds, presolve used 0,002 seconds,
       ... 7,851 seconds in simplex solver, in total 21,394 seconds.
```

Figura 66: Resultat de lpsolve pel conjunt de restriccions

enable(1,T251)&silent(1,T252)&fire(2,T400)&ntokens(3,2,P251)&cyclicsolution(4)&
nfire(4,T35,2)&fire(5,T300)&disable(6,T15)&fire(6,T0)&silent(7,T300)&fire(7,T301)&
[cyclicsolution(8)&¬fire(8,T350)]=>fire(7,T350)
&enable(9,T25)&enable(9,T325)&nfire(10,T0,2)&fire(10,T499)

6.2 Multiprocessador

El model que es mostra a continuació representa un sistema multiprocessador amb 5 processos i 2 busos disponibles on cada lloc simbolitza:

- P0, processos actius
- P1, busos disponibles
- P2, processos en espera
- P3, processos accedint als recursos
- P4, processos encuats esperant memòria comú

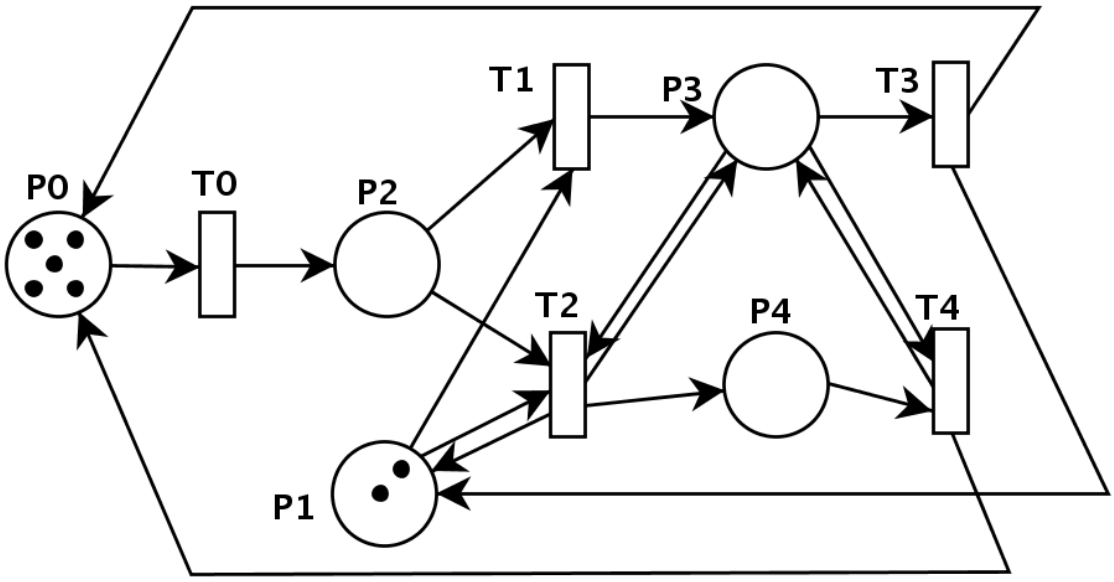


Figura 67: XdP que model un sistema multiprocessador

6.2.1 Dues traces

Comprovem primer un cas senzill en que es llencen dos processos i al cap d'una estona es llencen dos processos més. Verificarem un sistema amb dues traces on volem veure que si tenim una solució final cíclica, els processos que no s'executin a la primera traça ho faran a la segona.

La propietat que es verifica és la següent:

```
[[nfire(1,T0,2)&nfire(2,T0,2)&enable(1,T4)&cyclicsolution(2)]=>fire(2,T4)]&
[[ntokens(1,1,P3)&silent(1,T3)&ntokens(2,5,P0)]=>fire(2,T3)]
```

El primer pas és carregar el fitxer contingut amb la XdP. El sistema el valida, en crea l'estructura de dades associada i genera la seva imatge, que té el següent aspecte:

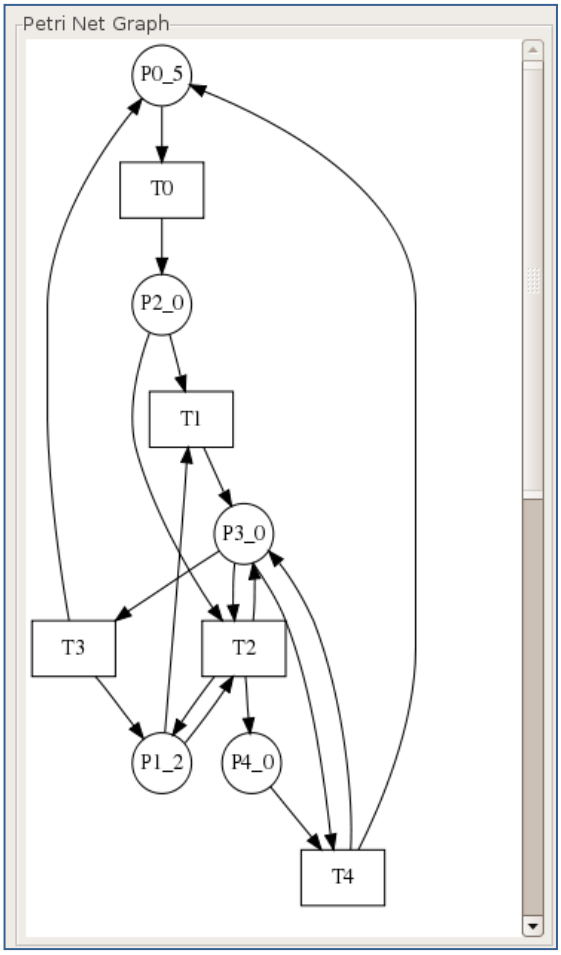


Figura 68: Imatge de la XdP que modela un multiprocessador generada pel sistema a partir d'un fitxer .dot

Donat que la xarxa és vàlida, la imatge es carrega automàticament, sense necessitat de generar cap missatge. A continuació s'introdueixen les propietats:

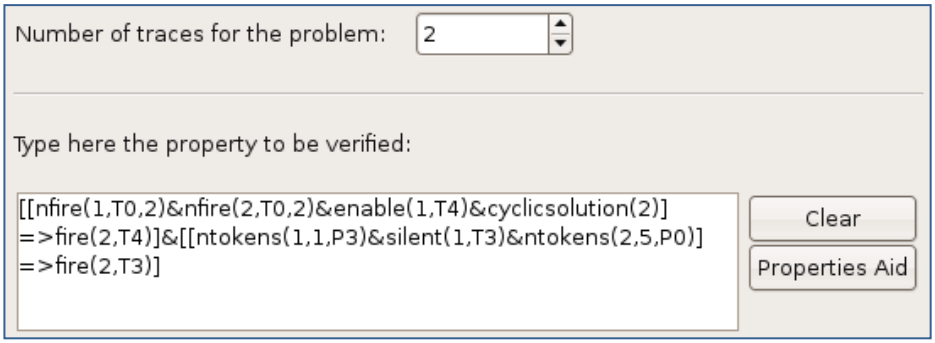


Figura 69: Aspecte de la part de Propietats de la interfície gràfica per tal de verificar les propietats indicades per dues traces

L’arbre generat per PCCTS en el seu recorregut en inordre és el que es mostra a continuació. S’indica quins són els conjunts mínims d’execució que es calcularan al solucionador:

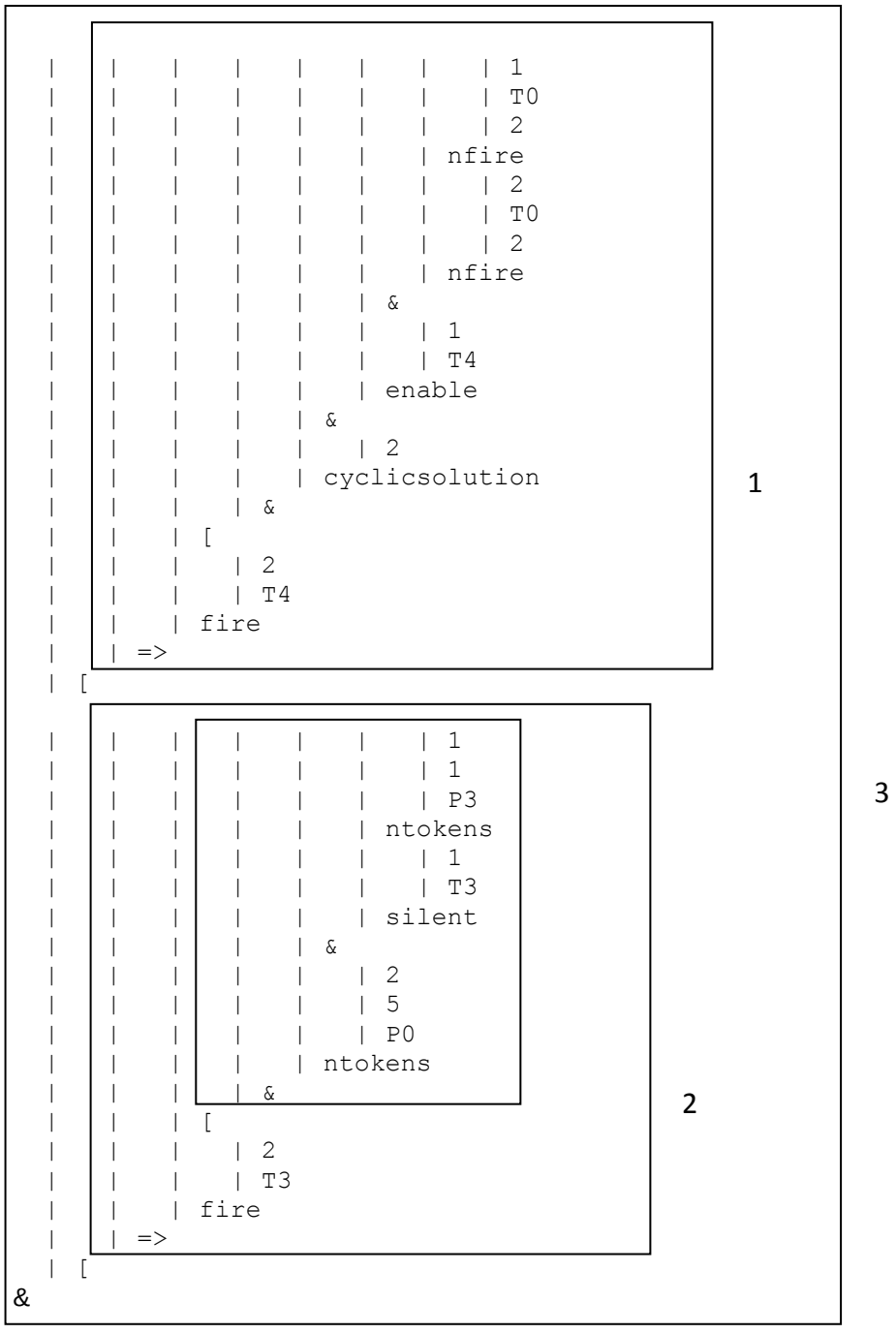


Figura 70: Arbre generat per la gramàtica amb les propietats.
Es soluciona d’esquerra a dreta solucionant els conjunts mínims:

1. Primer la implicació amb etiqueta “1” degut als claudàtors.
2. Segon la implicació amb etiqueta “2”, pel mateix fet.
3. Finalment, la concatenació de totes les propietats

Finalment es mostra, per a cada model (grup mínim d'execució), les dades calculades i el resultat obtingut:

```
        C1   C2   C3   C4   C5   C6   C7   C8   C9  C10
  Min    0    0    0    0    0    0    0    0    0    0

 R1    -1    0    0    1    1    0    0    0    0    0 >=   -5
 R2     0   -1    0    1    0    0    0    0    0    0 >=   -2
 R3     1   -1   -1    0    0    0    0    0    0    0 >=    0
 R4     0    1    0   -1    0    0    0    0    0    0 >=    0
 R5     0    0    1    0   -1    0    0    0    0    0 >=    0
 R6    -1    0    0    1    1   -1    0    0    1    1 >=   -5
 R7     0   -1    0    1    0    0   -1    0    1    0 >=   -2
 R8     1   -1   -1    0    0    1   -1   -1    0    0 >=    0
 R9     0    1    0   -1    0    0    1    0   -1    0 >=    0
 R10    0    0    1    0   -1    0    0    1    0   -1 >=    0

 R11   -1    0    0    1    1   -1    0    0    1    1  =    0
 R12    0   -1    0    1    0    0   -1    0    1    0  =    0
 R13    1   -1   -1    0    0    1   -1   -1    0    0  =    0
 R14    0    1    0   -1    0    0    1    0   -1    0  =    0
 R15    0    0    1    0   -1    0    0    1    0   -1  =    0

 R16   -1    0    0    1    1    0    0    0    0    0 >=   -5
 R17    0   -1    0    1    0    0    0    0    0    0 >=   -2
 R18    1   -1   -1    0    0    0    0    0    0    0 >=    0
 R19    0    1    0   -1    0    0    0    0    0    0 >=    1
 R20    0    0    1    0   -1    0    0    0    0    0 >=    1

 Type   R    R    R    R    R    R    R    R    R    R
 upbo   2  Inf  Inf  Inf  Inf    2  Inf  Inf  Inf    0
 lowbo  2    0    0    0    0    2    0    0    0    0
```

A

B

C

D

```
Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size: 20 constraints, 10 variables,  72 non-zeros.
Sets:        0 GUB,     0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

The model is INFEASIBLE
lp_solve unsuccessful after 7 iter and a last best value of 1e+30
MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 7, 0 (0,0%) were bound flips.
      There were 0 refactorizations, 0 triggered by time and 0 by density.
       ... on average 7,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 21 NZ entries, 1,0x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,002 seconds, presolve used 0,000 seconds,
       ... 0,001 seconds in simplex solver, in total 0,003 seconds.
```

Figura 71: Model solucionat per lpsolve pel conjunt de propietats "1" de la Figura 70

Els grups d'instruccions indicats corresponen a:

- Restricció per defecte del sistema: $M_0 + Ax \geq 0$ per les dues traces.
- Ciclicitat per la traça 2: $Ax = 0$.
- Activació de la transició T4 a la traça 1.
- Limitacions de columna: nombre de dispars de les transicions.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10		
Min	0	0	0	0	0	0	0	0	0	0		
R1	-1	0	0	1	1	0	0	0	0	0	>=	-5
R2	0	-1	0	1	0	0	0	0	0	0	>=	-2
R3	1	-1	-1	0	0	0	0	0	0	0	>=	0
R4	0	1	0	-1	0	0	0	0	0	0	>=	0
R5	0	0	1	0	-1	0	0	0	0	0	>=	0
R6	-1	0	0	1	1	-1	0	0	1	1	>=	-5
R7	0	-1	0	1	0	0	-1	0	1	0	>=	-2
R8	1	-1	-1	0	0	1	-1	-1	0	0	>=	0
R9	0	1	0	-1	0	0	1	0	-1	0	>=	0
R10	0	0	1	0	-1	0	0	1	0	-1	>=	0
R11	-1	0	0	1	1	-1	0	0	1	1	=	0
R12	0	-1	0	1	0	0	-1	0	1	0	>=	-2
R13	1	-1	-1	0	0	1	-1	-1	0	0	>=	0
R14	0	1	0	-1	0	0	1	0	-1	0	>=	0
R15	0	0	1	0	-1	0	0	1	0	-1	>=	0
R16	-1	0	0	1	1	0	0	0	0	0	>=	-5
R17	0	-1	0	1	0	0	0	0	0	0	>=	-2
R18	1	-1	-1	0	0	0	0	0	0	0	>=	0
R19	0	1	0	-1	0	0	0	0	0	0	=	1
R20	0	0	1	0	-1	0	0	0	0	0	>=	0
Type	R	R	R	R	R	R	R	R	R	R		
upbo	2	Inf	Inf	Inf	Inf	2	Inf	Inf	Inf	0		
lowbo	2	0	0	0	0	2	0	0	0	0		

A

E

F

G

```
Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size: 20 constraints, 10 variables,  72 non-zeros.
Sets:        0 GUB,     0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

The model is INFEASIBLE
lp_solve unsuccessful after 7 iter and a last best value of 1e+30
MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 7, 0 (0,0%) were bound flips.
      There were 0 refactorizations, 0 triggered by time and 0 by density.
       ... on average 7,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 21 NZ entries, 1,0x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,002 seconds, presolve used 0,000 seconds,
       ... 0,001 seconds in simplex solver, in total 0,003 seconds.
```

Figura 72: Model pel conjunt de propietats "2" de la Figura 70

Els grups d'instruccions indicats corresponen a:

- Restricció per defecte del sistema: $M_0 + Ax \geq 0$ per les dues traces.
- Nombre de marques a P0 al final de la traça 2: $M_{00} + Ax_0 = 5 - 5$.
- Nombre de marques a P3 al final de la traça 1: $M_{03} + Ax_3 = 1 - 5$.
- Limitacions de columna: nombre de dispars de les transicions.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10		
Min	0	0	0	0	0	0	0	0	0	0		
R1	-1	0	0	1	1	0	0	0	0	0	>=	-5
R2	0	-1	0	1	0	0	0	0	0	0	>=	-2
R3	1	-1	-1	0	0	0	0	0	0	0	>=	0
R4	0	1	0	-1	0	0	0	0	0	0	>=	0
R5	0	0	1	0	-1	0	0	0	0	0	>=	0
R6	-1	0	0	1	1	-1	0	0	1	1	>=	-5
R7	0	-1	0	1	0	0	-1	0	1	0	>=	-2
R8	1	-1	-1	0	0	1	-1	-1	0	0	>=	0
R9	0	1	0	-1	0	0	1	0	-1	0	>=	0
R10	0	0	1	0	-1	0	0	1	0	-1	>=	0

A

R11	-1	0	0	1	1	-1	0	0	1	1	=	0
R12	0	-1	0	1	0	0	-1	0	1	0	=	0
R13	1	-1	-1	0	0	1	-1	-1	0	0	=	0
R14	0	1	0	-1	0	0	1	0	-1	0	=	0
R15	0	0	1	0	-1	0	0	1	0	-1	=	0

B

R16	-1	0	0	1	1	0	0	0	0	0	>=	-5
R17	0	-1	0	1	0	0	0	0	0	0	>=	-2
R18	1	-1	-1	0	0	0	0	0	0	0	>=	0
R19	0	1	0	-1	0	0	0	0	0	0	>=	1
R20	0	0	1	0	-1	0	0	0	0	0	>=	1

C

R21	-1	0	0	1	1	-1	0	0	1	1	=	0
R22	0	-1	0	1	0	0	-1	0	1	0	>=	-2
R23	1	-1	-1	0	0	1	-1	-1	0	0	>=	0
R24	0	1	0	-1	0	0	1	0	-1	0	>=	0
R25	0	0	1	0	-1	0	0	1	0	-1	>=	0

E

R26	-1	0	0	1	1	0	0	0	0	0	>=	-5
R27	0	-1	0	1	0	0	0	0	0	0	>=	-2
R28	1	-1	-1	0	0	0	0	0	0	0	>=	0
R29	0	1	0	-1	0	0	0	0	0	0	=	1
R30	0	0	1	0	-1	0	0	0	0	0	>=	0

F

Type	R	R	R	R	R	R	R	R	R	R
upbo	2	Inf	Inf	Inf	Inf	2	Inf	Inf	Inf	0
lowbo	2	0	0	0	0	2	0	0	0	0

H

```
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

bfp_factorize: Resolving 1 singularity at refact 1, iter 0
bfp_factorize: Resolving 1 singularity at refact 1, iter 0

Value of objective function: 0
Actual values of the variables:
C1                2
C2                1
C3                1
C4                0
C5                0
C6                2
C7                2
C8                0
C9                3
C10               1
Optimal solution              0 after     6 iter.
Excellent numeric accuracy ||*|| = 0
```

```
MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 6, 0 (0,0%) were bound flips.
      There were 1 refactorizations, 0 triggered by time and 0 by density.
       ... on average 6,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 36 NZ entries, 0,7x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,007 seconds, presolve used 0,000 seconds,
       ... 0,001 seconds in simplex solver, in total 0,008 seconds.
```

Figura 73: Model pel conjunt de propietats "3" de la Figura 70

Els grups d'instruccions indicats corresponen a:

- A, B, C, E i F es corresponen amb les anteriors models.
- H: restriccions de columna on la part dreta de les antigues implicacions ara ja no està negada, per que són concatenacions.

Al final les propietats s'han complert, i el resultat que es reporta és el vector de comptatge de dispars que facilita aquesta solució:

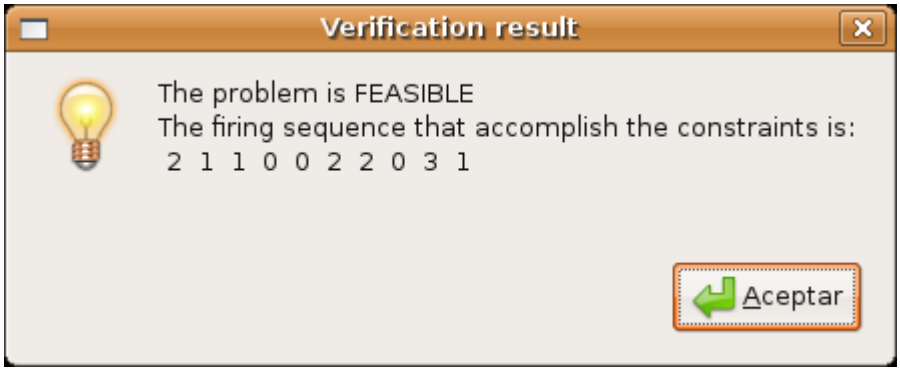


Figura 64: Resultat reportat per la interfície gràfica

6.2.2 Quatre traces

Anem a comprovar ara un conjunt de propietats sobre la mateixa XdP, però per a quatre traces. Estenem el conjunt de propietats verificat anteriorment amb propietats a verificar per les traces 3 i 4. La propietat queda com:

[[nfire(1,T0,2)&nfire(2,T0,2)&enable(1,T4)&cyclicsolution(2)]=>fire(2,T4)]&

[[ntokens(1,1,P3)&silent(1,T3)&ntokens(2,5,P0)]=>fire(2,T3)]&

[nfire(3,T0,2)&¬silent(3,T3)&silent(3,T4)]&

[cyclicsolution(4)=>enable(4,T0)]&

[nfire(4,T0,2)&fire(4,T4)&silent(4,T3)=>fire(4,T2)]

Ara el model a resoldre és més gran. L'arbre generat és el següent:

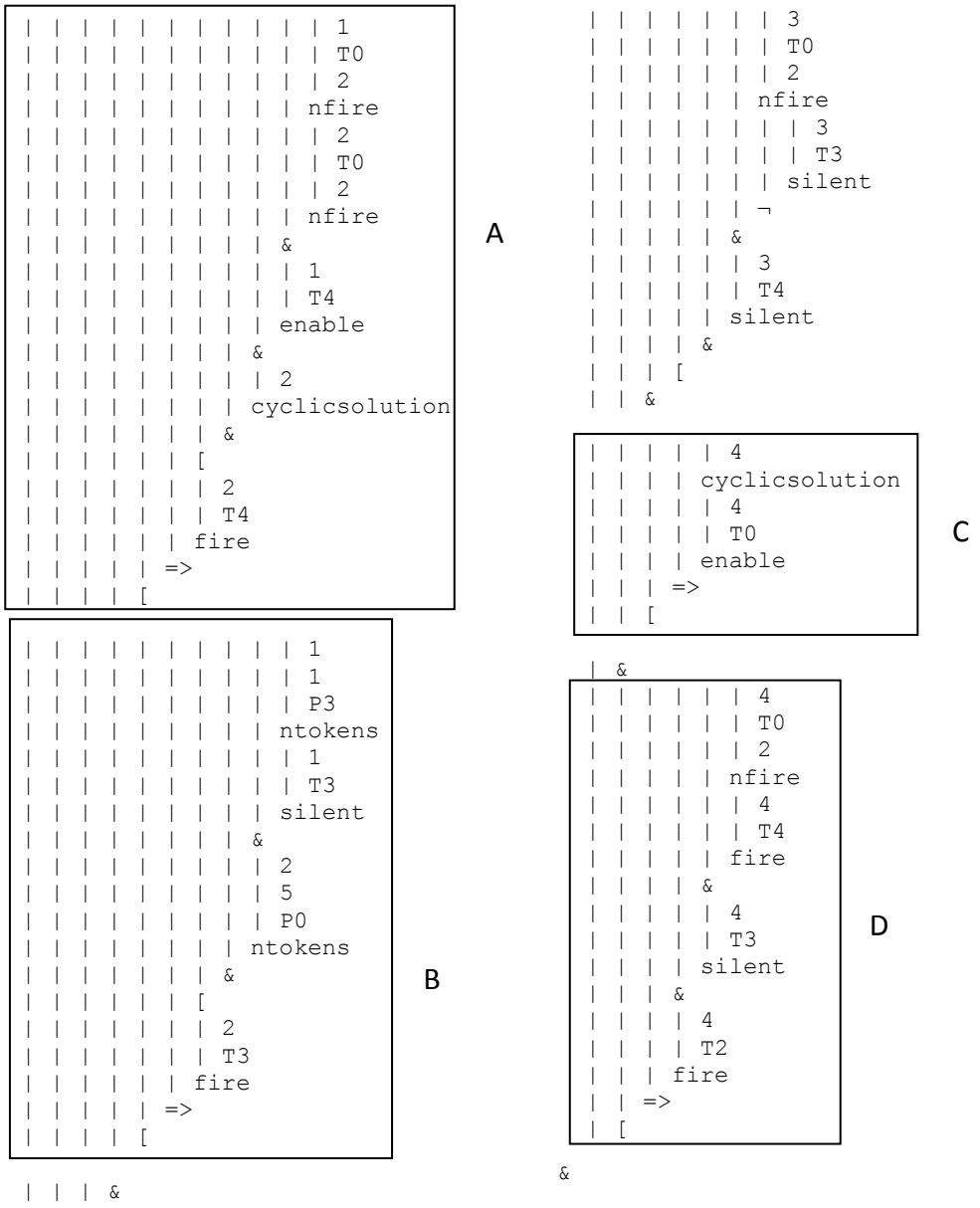


Figura 65: Arbre generat per la gramàtica amb 5 grups de propietats:

1. Els dos primers, A i B, són els mateixos que per l'exemple de dues traces, així que no posarem el model de PL a continuació.
2. Hi ha un tercer grup que no s'executa per sí sol per que, en estar relacionat per operacions de concatenació, el sistema va corregint la seva prioritat fins executar totes les propietats juntes.
3. Els grups C i D, que també s'executen independentment.
4. Finalment, s'executa tot el model essent tots els operadors conjuncions.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20		
Min	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
R1	-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	-5
R2	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	-2
R3	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R4	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R5	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R6	-1	0	0	1	1	-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	>=	-5
R7	0	-1	0	1	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	>=	-2
R8	1	-1	-1	0	0	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R9	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	>=	0
R10	0	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	>=	0
R11	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	0	0	0	0	0	>=	-5
R12	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	0	0	0	0	0	>=	-2
R13	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	0	0	0	0	0	>=	0
R14	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	>=	0
R15	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	>=	0
R16	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	>=	-5
R17	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	>=	-2
R18	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	>=	0
R19	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	>=	0
R20	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	>=	0
R21	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	>=	-4
R22	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	>=	-2
R23	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	>=	0
R24	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	>=	0
R25	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	>=	0
R26	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	-1	0	0	1	1	=	0
R27	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	0	-1	0	1	0	=	0
R28	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	1	-1	-1	0	0	=	0
R29	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	=	0
R30	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	0	0	1	0	-1	=	0
R31	-1	0	0	1	1	-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	=	0
R32	0	-1	0	1	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	>=	-2
R33	1	-1	-1	0	0	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R34	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	>=	0
R35	0	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	>=	0
R36	-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	-5
R37	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	-2
R38	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R39	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=	1
R40	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R41	-1	0	0	1	1	-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	=	0
R42	0	-1	0	1	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	=	0
R43	1	-1	-1	0	0	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	=	0
R44	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	=	0
R45	0	0	1	0	-1	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	=	0
R46	-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	-5
R47	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	-2
R48	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	0
R49	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	1
R50	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	>=	1
Type	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R		
upbo	2	Inf	Inf	0	Inf	2	Inf	Inf	Inf	Inf	2	Inf	Inf	Inf	0	2	Inf	Inf	0	Inf		
lowbo		2	0	0	0	0	2	0	0	1	1	2	0	0	1	0	2	0	1	0		

1 2 3 4 5 6 7 8

Model final:

Els requadres indicats referencien, per ordre de dalt a baix:

1. Restriccions per defecte: solució positiva.
2. La transició T0 ha d'estar activa al final de la traça 4.
3. La solució per la traça 4 ha de ser cíclica.
4. El nombre de marques al final de la traça 2 pel lloc P0 ha de ser 5.
5. El nombre de marques al final de la traça 1 pel lloc P3 ha de ser 1.
6. La solució per la traça 2 ha de ser cíclica.
7. La transició 4 ha d'estar activa al final de la traça 1.
8. Conjunt de restriccions de columna que es donen per a les 4 traces.

```
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Value of objective function: 0
Actual values of the variables:
C1                2
C2                1
C3                1
C4                0
C5                0
C6                2
C7                2
C8                0
C9                3
C10               1
C11               2
C12               2
C13               0
C14               2
C15               0
C16               2
C17               0
C18               2
C19               0
C20               2
Optimal solution              0 after     8 iter.
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 8, 0 (0,0%) were bound flips.
      There were 1 refactorizations, 0 triggered by time and 0 by density.
       ... on average 8,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 201 NZ entries, 1,0x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,009 seconds, presolve used 0,000 seconds,
           ... 0,000 seconds in simplex solver, in total 0,009 seconds.
```

Figura 66: Solució de lpsolve pel conjunt de propietats

[[nfire(1,T0,2)&nfire(2,T0,2)&enable(1,T4)&cyclicsolution(2)]=>fire(2,T4)]&
[[ntokens(1,1,P3)&silent(1,T3)&ntokens(2,5,P0)]=>fire(2,T3)]&
[nfire(3,T0,2)&¬silent(3,T3)&silent(3,T4)]&
[cyclicsolution(4)=>enable(4,T0)]&
[nfire(4,T0,2)&fire(4,T4)&silent(4,T3)=>fire(4,T2)]

6.3 Solució entera o real

Tal i com s'ha vist a teoria, és possible que l'equació de marcatge tingui solució al domini dels nombres Racionals però que no en tingui al dels Enters. L'àmbit dels enters és més restrictiu.

Prenem com a base d'estudi la XdP que es mostra a continuació:

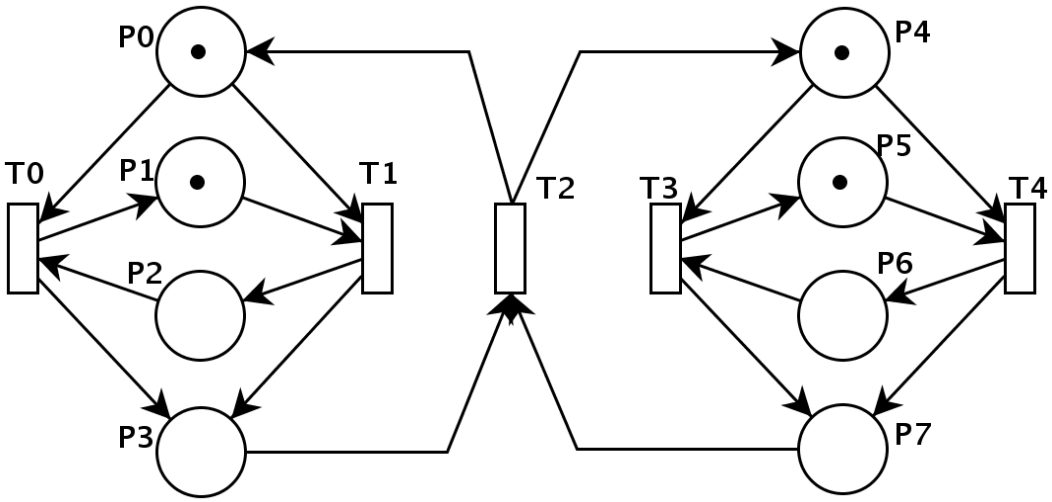


Figura 67: Xarxa de Petri

Comprovem si el marcatge $M_0 = (1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)$ té solució per la xarxa. La propietat a verificar és *reachable(1,1,0,1,0,1,1,0,0)*. El model genera les el següent sistema d'inequacions:

	C1	C2	C3	C4	C5		
Minimize	0	0	0	0	0		
R1	-1	-1	1	0	0	>=	-1
R2	1	-1	0	0	0	>=	-1
R3	-1	1	0	0	0	>=	0
R4	1	1	-1	0	0	>=	0
R5	0	0	1	-1	-1	>=	-1
R6	1	0	0	1	-1	>=	-1
R7	0	0	0	-1	1	>=	0
R8	0	0	-1	1	1	>=	0
R9	-1	-1	1	0	0	=	0
R10	1	-1	0	0	0	=	-1
R11	-1	1	0	0	0	=	1
R12	1	1	-1	0	0	=	0
R13	0	0	1	-1	-1	=	0
R14	0	0	0	1	-1	=	0
R15	0	0	0	-1	1	=	0
R16	0	0	-1	1	1	=	0
Type	Real	Real	Real	Real	Real		
upbo	Inf	Inf	Inf	Inf	Inf		
lowbo	0	0	0	0	0		

, on es representen les restriccions:

- La solució per totes les variables ha de ser positiu, $M_0 + Ax \geq 0$ [de R1 a R5].
- El marcatge final ha de ser $(1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)$: $M_0 + Ax = M$ [de R9 a R16].

La solució que dóna la llibreria és:

```
Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:        16 constraints,        5 variables,       40 non-
zeros.
Sets:         0 GUB,        0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Value of objective function: 0

Actual values of the variables:
C1          0
C2          1
C3          1
C4        0,5
C5        0,5

Optimal solution   0 after     4 iter.

Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 4, 0 (0,0%) were bound flips.
      There were 0 refactorizations, 0 triggered by time and 0 by density.
       ... on average 4,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 17 NZ entries, 1,0x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,002 seconds, presolve used 0,000 seconds,
       ... 0,002 seconds in simplex solver, in total 0,004 seconds.
```

Figura 68: Solució de lpsolve per la restricció "reachable(1,1,0,1,0,1,1,0,0)"

El solucionador retorna el vector de parik $\sigma = \left(0 \quad 1 \quad 1 \quad \frac{1}{2} \quad \frac{1}{2}\right)$.

Provem ara de trobar una solució en l'àmbit dels Enters. Per això caldrà afegir una nova restricció al model, i en conjunt les propietats a verificar quedarien com *reachable(1,1,0,1,0,1,1,0,0)&intsolution(1)*. De la taula anterior únicament canvia la fila etiquetada com "Type" que, en comptes de tenir, per a cada columna, el valor "Real", tindrà "Int". I en aquest cas lpsolve no és capaç de trobar solució:

```
Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:   16 constraints,  5 variables,   40 non-zeros.

Sets:     0 GUB,     0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Relaxed solution       0 after      4 iter is B&B base.

lp_solve unsuccessful after 104 iter and a last best value of 1e+30
lp_solve explored 200 nodes before termination

MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 104, 0 (0,0%) were bound flips.
      There were 101 refactorizations, 0 triggered by time and 1 by density.
       ... on average 1,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 45 NZ entries, 1,0x largest
      basis.
      The maximum B&B level was 101, 10,1x MIP order, with 200 nodes
      explored.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,001 seconds, presolve used 0,000 seconds,
       ... 0,009 seconds in simplex solver, in total 0,010 seconds.
```

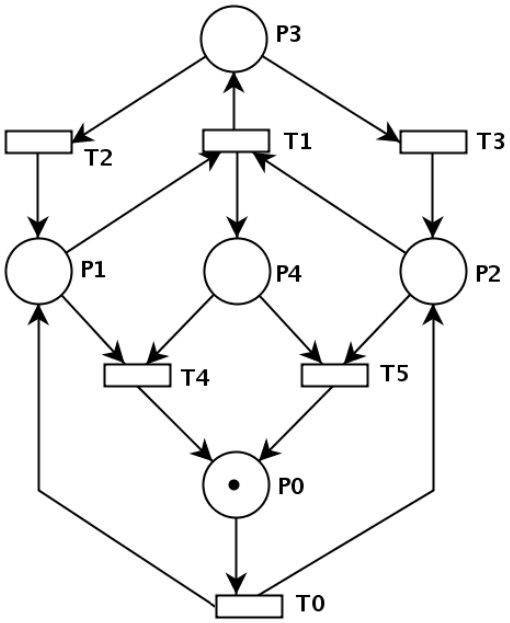
Figura 69: Solució de lpsolve per la restricció "reachable(1,1,0,1,0,1,1,0,0)&intsolution(1)"

6.4 Resultats espuris

L'equació de marcatge és una condició necessària però no suficient per a verificar que un marcatge m és abastable a partir d'un marcatge m_0. Existeixen limitacions que queden definides a l'apartat d'aquest document. És per a això que a continuació es mostra diferents exemples en què les restriccions extretes de l'equació de marcatge aplicades a un model de PL que no són aplicables a la XdP estudiada.

6.4.1 Marcatges negatius

La xarxa que es mostra en aquest exemple està extreta de l'article "Synthesis of Asynchronous Controllers using Integer Linear Programing" (5).



$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & -1 \end{pmatrix}$$

$$M_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$$

Figura 70: XdP amb la seva matriu d'incidència i el marcatge inicial

Per aquesta XdP volem verificar que el model de PL pot donar un resultat que implica una seqüència de dispars on, en algun moment, es passa per un estat de la XdP on hi ha algun lloc amb un nombre de marques negatiu. Això és quelcom que no volem com a solució per la XdP, ja que no és possible en un sistema real.

Provem de verificar la restricció: $M = (0 \quad 0 \quad 1 \quad 1 \quad 0)$ per $M_0 + Ax = M$. Això genera el model de PL:

	C1	C2	C3	C4	C5	C6		
Minimize	0	0	0	0	0	0		
R1	-1	0	0	0	1	1	>=	-1
R2	1	-1	1	0	-1	0	>=	0
R3	1	-1	0	1	0	1	>=	0
R4	0	1	-1	-1	0	0	>=	0
R5	0	1	0	0	-1	-1	>=	0
R6	-1	0	0	0	1	1	=	-1
R7	1	-1	1	0	-1	0	=	0
R8	1	-1	0	1	0	-1	=	1
R9	0	1	-1	-1	0	0	=	1
R10	0	1	0	0	-1	-1	=	0
Type	Real	Real	Real	Real	Real	Real		
upbo	Inf	Inf	Inf	Inf	Inf	Inf		
lowbo	0	0	0	0	0	0		

, on es representen les restriccions:

- La solució per totes les variables ha de ser positiu, $M_0 + Ax \geq 0$ [de R1 a R5].
- El marcatge final ha de ser $(0 \quad 0 \quad 1 \quad 1 \quad 0)$: $M_0 + Ax = M$ [de R6 a R10].

Com a resultat del solucionador lpsolve s'obté:

```
Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:   10 constraints,    6 variables,     34 non-zeros.
Sets:     0 GUB,        0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Value of objective function: 0

Actual values of the variables:
C1          2
C2          1
C3          0
C4          0
C5          1
C6          0

Optimal solution   0 after     3 iter.

Excellent numeric accuracy ||*|| = 0
MEMO: lp_solve version 5.5.0.15 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 3, 0 (0,0%) were bound flips.
      There were 0 refactorizations, 0 triggered by time and 0 by density.
       ... on average 3,0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 11 NZ entries, 1,0x largest
      basis.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0,017 seconds, presolve used 0,000 seconds,
       ... 0,000 seconds in simplex solver, in total 0,017 seconds.
```

Figura 71: Resultat de lpsolve per a la restricció reachable(0,0,1,1,0)

Aquesta solució indica que el marcatge $M = (0 \quad 0 \quad 1 \quad 1 \quad 0)$ és abastable a partir del marcatge inicial $M_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$ per un conjunt de dispars representats pel vector de parik $\sigma = (2 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0)$.

Tot i això, el marcatge M no és abastable a partir de cap seqüència de dispars factible, només es pot arribar a M passant per seqüències de dispars que visiten marcatges negatius, com per exemple: $(1 \quad 0 \quad 0 \quad 0 \quad 1) \xrightarrow{T_0} (0 \quad 1 \quad 1 \quad 0 \quad 0) \xrightarrow{T_1} (0 \quad 0 \quad 0 \quad 1 \quad 1) \xrightarrow{T_4} (1 \quad \mathbf{-1} \quad 0 \quad 1 \quad 0) \xrightarrow{T_0} (0 \quad 0 \quad 1 \quad 1 \quad 0)$.

M és un resultat espuri de l'equació de marcatge. Per la XdP donada existeix un espai de resultats espuris que donen solució per l'equació de marcatge.

A la següent imatge es mostra el graf d'abastabilitat de la xarxa i el conjunt de resultats espuris:

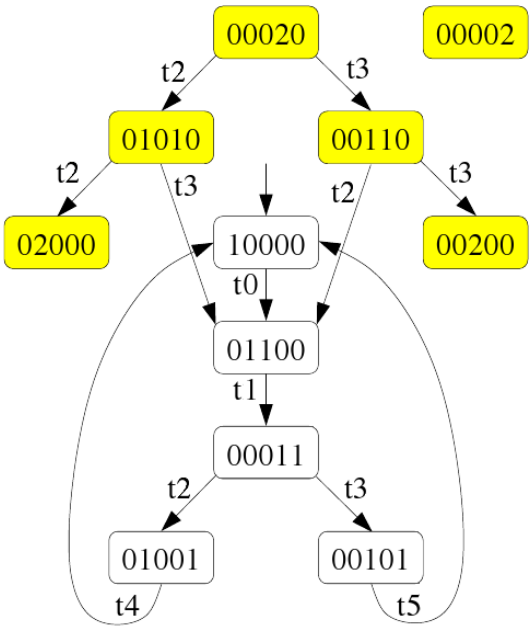


Figura 72: Graf d'abastabilitat de la XdP de la Figura 70 amb un conjunt de resultats espuris que donen solució per l'equació de marcatge

6.4.2 Enriquiment amb invariants

Tal i com s'ha comentat, l'eina permet verificar si un vector és un invariant d'una XdP. El que es vol en aquest punt és comprovar l'eficàcia de l'ús d'invariants per a enriquir la propietat d'abastabilitat que està implementada a l'eina.

Seguin amb l'exemple anterior, comprovem ara la següent propietat:

reachable(1,0,2,0,0,0)&sinvariant(2,1,1,1,1)

que verifica si el vector $M = (0 \quad 2 \quad 0 \quad 0 \quad 0)$ és abastable al final de la traça 1 i si el vector $x = (2 \quad 1 \quad 1 \quad 1 \quad 1)$ és un S-invariant del sistema.

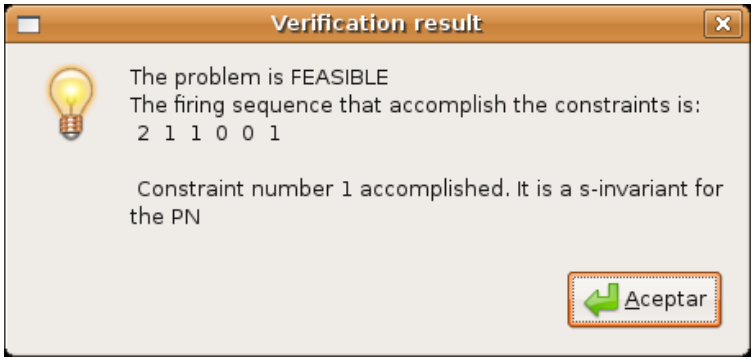


Figura 73: Resultat que retorna la interfície després que l'eina resolgui el model "reachable(1,0,2,0,0,0)&sinvariant(2,1,1,1,1)"

Així doncs, segons el model actual, el marcatge és abastable.

Volem utilitzar la condició necessària dels S-invariants sobre la propietat d'abastabilitat, i comprovem si $M_0 \cdot x = M \cdot x$:

$$\left.\begin{array}{l} M_0 \cdot x = (1 \quad 0 \quad 0 \quad 0 \quad 0)\cdot(2 \quad 1 \quad 1 \quad 1 \quad 1) = 2. \\ M \cdot x = (0 \quad 2 \quad 0 \quad 0 \quad 0)\cdot(2 \quad 1 \quad 1 \quad 1 \quad 1) = 2. \end{array}\right\} \quad 2 = 2$$

Es compleix la igualtat $M_0 \cdot x = M \cdot x$, així que la condició necessària dels S-invariants, en aquest cas, no ens ha aportat més informació que l'equació de marcatge.

Tanmateix, sí que ens aporta informació sobre la limitació, ja que en verificar que existeix un S-invariant a la xarxa, $x = (2 \quad 1 \quad 1 \quad 1 \quad 1)$ es verifica que la xarxa és **limitada**.

Amb la eina PIPE2, esmentada a la part de Implementació (apartat 5.1.2 d'aquest document), hem trobat quins són els S-invariants de la XdP de la Figura 70. El vector $x = (2 \quad 1 \quad 1 \quad 1 \quad 1)$ n'és l'únic.

Podem dir la xarxa és viva per al marcatge inicial $M_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$ si es compleix $I \cdot M_0 > 0$ per tot S-invariant semi-positiu:

$$(2 \quad 1 \quad 1 \quad 1 \quad 1) \cdot (1 \quad 0 \quad 0 \quad 0 \quad 0) = 2 > 0$$

La xarxa és **viva** pel marcatge inicial indicat.

6.4.3 Enriquiment amb predicats estables

Tot i que l'eina no implementa cap tipus de verificació respecte dels predicats estables, sí que se'n fa referència a l' "Annex 1" per que es considera una bona tècnica per ser aplicada en futures ampliacions per enriquir els resultats de l'equació de marcatge.

En aquest sentit, seguim amb l'exemple introduït a l'apartat anterior, amb l'enriquiment amb invariants (XdP de la Figura 70) per veure si aplicant les propietats dels traps o dels siphons millorem la solució.

Aplicarem el Apliquem el **test de no-abastabilitat dels traps**. Amb l'eina PIPE2 trobem els traps de la xarxa: *{P0, P2, P4}, {P0, P1, P4}, {P0, P1, P2, P3}*. El test consisteix a verificar que si el marcatge inicial M_0 marca al menys un lloc d'un trap mentre que el marcatge final M no en marca cap, llavors el M no és abastable des de M_0. Comprovem els casos:

- $M_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$ marca lloc a tots els traps: tots són candidats.
- $M = (0 \quad 2 \quad 0 \quad 0 \quad 0)$ no marca cap lloc del trap *{P0, P2, P4}*.

Gràcies a aquest test em pogut concloure, com ja sabíem però ni l'equació de marcatge ni el test dels S-invariants han arribat a decidir, que el marcatge $M = (0 \quad 2 \quad 0 \quad 0 \quad 0)$ **no és abastable** des de $M_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$.

7 Planificació i Costs

7.1.1 Planificació

La planificació final ha acabat diferent de la que es va pensar inicialment degut a una concepció massa benvolent del cost que totes les parts s'integressin en una mateixa eina i al desconeixement de moltes de les eines utilitzades.

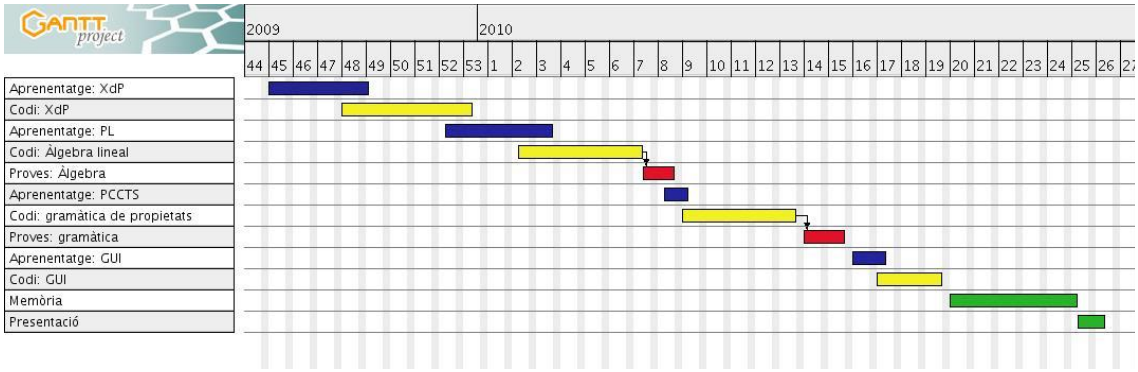


Figura 74: Diagrama de Gantt amb la planificació final del projecte

Tot i que en la planificació inicial es va pensar en afegir a la eina tècniques d'anàlisi de Xarxes de Petri de les que es parla a l' "Annex 1" no ha sigut possible incloure-les per qüestió de temps. El desavantatge amb el llenguatge de programació, molt oblidat, i el desconeixement de molta teoria aplicada en aquest projecte han fet que cada part s'hagi allargat en una setmana i al final, el temps esperat per aplicar al menys invariants en el projecte no ha sigut el que s'esperava.

7.1.2 Càrrega de treball

El projecte es va començar a planificar al juny del 2009, però el seu desenvolupament no es va iniciar fins al novembre 2009. La durada ha estat de 8 mesos, des del novembre de 2009 fins al juliol de 2010.

S'ha realitzat les tasques d'aprenentatge sobre la matèria, l'anàlisi, el disseny i la implementació entre el novembre de 2009 i mitjans de maig de 2010 amb una dedicació diària de 4 hores. Descomptant festius en total han estat 136 dies de feina.

La memòria s'ha realitzat entre el mes de maig i finals de juny del 2010, quatre dies a la setmana, unes 4 hores al dia. En total s'ha dedicat 31 dies a la realització de la memòria.

Concepte	Hores
Projecte	546
Documentació	147
TOTAL	**693**

7.1.3 Cost del projecte

El programari utilitzat durant el desenvolupament d'aquest projecte es de lliure distribució així que el seu cost ha estat zero.

El cost total del projecte es calcula respecte del nombre d'hores dedicades segons les tasques realitzades i el hardware utilitzat.

Etapa	**Rol**	**Preu/hora**	**Hores**	**Cost**
Aprenentatge	Analista	35€	50	1.750
Desenvolupament	Programador	30€	420	12.600
Proves	Analista/programador	30€-35€	76	2.470
TOTAL			546	16.820

El hardware utilitzat ha estat un ordinador portàtil: 900€.

El cost total és de **17.720€**.

8 Conclusió

En aquest punt s'exposa les diferents valoracions del projecte. D'una banda les conclusions extretes sobre la realització del projecte i la seva aportació a la matèria que es tracta; juntament a aquesta valoració s'inclou l'anàlisi dels objectius assolits i la feina futura. A continuació es fa una valoració del cost, tant temporal com financer, del projecte. Finalment es dóna una valoració personal de la feina realitzada.

8.1 Conclusions generals

El que es presenta en aquest projecte és una aplicació pràctica basada en Programació Lineal de mètodes que consisteixen en l'ús de l'equació de marcatge per l'anàlisi de Xarxes de Petri.

Una Xarxa de Petri com a model d'un sistema esta formada per la seva part estructural (els llocs i les transicions) i el seu marcatge (que es modifica segons els diferents estats). Nosaltres hem treballat amb xarxes marcades ordinàries (els pes dels arcs és sempre 1) i ens hem centrat en estudis comportamentals a partir la descripció d'una sèrie de propietats generals i uns comportaments específics de la Xarxa de Petri que es poden verificar usant la tècnica de l'anàlisi matricial.

S'ha inclòs part de la teoria d'invariants, permetent també la verificació d'aquest tipus de vectors. Però degut a les limitacions temporals del projecte, ha quedat de banda l'aplicació d'aquesta tècnica per a la verificació de propietats.

En comptes d'això, s'ha creat una eina potent en quant a la possibilitat de relacionar diferents tipus de restriccions en una mateixa verificació. Permetent la negació, la concatenació i la implicació de propietats es posa a l'abast de l'usuari un ampli ventall de comportaments analitzables. S'estudia comportaments relatius a l'estat, com pot ser la concurrència i el conflicte de transicions, propietats comportamentals de la xarxa, com poden ser l'exclusió mútua, l'abastabilitat, i l'existència de solucions tenint en compte certes restriccions dins la xarxa, com poden ser la ciclicitat, l'activació o desactivació de transicions, el dispar o silenci de transicions i l'estat dels llocs. Aquestes propietats han estat escollides pel fet de poder aplicar tècniques d'àlgebra lineal sobre un únic model. Altres propietats com poden ser la vivacitat o la limitació, requereixen recórrer l'arbre d'abastabilitat per ser comprovades, o bé utilitzar altres tècniques a part de l'equació de marcatge.

S'ha comprovat l'eficàcia de l'eina a l'hora de tractar amb xarxes de Petri de mida mitjana i l'encert amb l'ús de Lpsolve com a solucionador de problemes de programació lineal. Ha

estat una eina fàcil d'adaptar a l'aplicació i que ha donat molts bons resultats en quant a consum de recursos i temps de càlcul. Com ja queda patent als articles referenciats i, també, després de comprovar-ho amb l'eina dissenyada, les tècniques de verificació basades en l'àlgebra lineal són els algoritmes més eficients.

Tanmateix, cal remarcar que durant l'aprenentatge s'ha vist, tal i com s'expressa a dins l'apartat de teoria 2.1.3.2.1.3 d'aquest document, que l'aplicació de l'equació de marcatge té certes limitacions, i això és degut a que solucions per a un sistema d'inequacions no han de ser forçosament solucions vàlides per a una seqüència de dispars d'una Xarxa de Petri. És per aquest fet que també s'ha implementat la verificació d'invariants i es parla a l' "Annex 1" d'altres tècniques que poden enriquir l'eina com són l'ús d'invariants i l'ús de predicats estables. El proposa fermament l'ús d'aquestes tècniques per enriquir l'eina implementada.

També cal remarcar l'ús de la llibreria GraphViz a l'hora de visualitzar les Xarxes de Petri des de la interfície gràfica. Per a xarxes molt grans, l'eina potser no les mostra de forma molt clara, però és una ajuda per l'usuari poder tenir visible la xarxa a l'hora d'analitzar-la i aquesta ha sigut una manera fàcil d'oferir aquesta possibilitat.

8.1.1 Objectius assolits

Els objectius d'aquest projecte estan assolits en gaire bé la seva totalitat.

S'ha desenvolupat, en base als objectius plantejats a l'inici del projecte, una eina amb interfície gràfica i una llibreria que permeten:

- Validar la correctesa d'una arxiu amb format XML que conté la descripció d'una XdP a partir d'un esquema predefinit basat en l'estàndard PNML.
- Validar un conjunt de propietats en base a la descripció feta amb una gramàtica de PCCTS.
- Crear un model de PL amb lpsolve a partir de les propietats definides i la XdP escollida, i verificar la seva solució.
- Retornar un resultat sobre cadascun dels passos realitzats.
- L'eina amb interfície gràfica, a més, permet visualitzar la XdP que s'està analitzant.

L'eina és fiable en cadascun dels seus passos tenint en compte les limitacions que ja s'ha explicat dels mètodes d'anàlisi utilitzats.

8.1.2 Treball futur

Aquest projecte s'ha realitzat com a base d'un projecte que pot ser molt més gran. Es consoliden els ciments d'un conjunt de tècniques demostrades sobre l'anàlisi de XdP i es planteja fermament l'ampliació de l'eina amb altres tècniques ja esmentades per enriquir el model de Programació Lineal. En aquest marc, les ampliacions que es proposa són:

- Incloure l'operador de disjunció a la gramàtica de validació de les propietats i amb això permetre verificacions que generin més d'un model. Amb aquesta millora es

permetria negar propietats que ara no és possible com assignar un nombre de marques a un lloc o relaxar la restricció de desactivació.

- Incloure la cerca d'invariants sobre una XdP i aplicar aquest coneixement per permetre verificar altres propietats com són la vivacitat i la limitació, i enriquir la restricció d'abastabilitat.
- Incloure la cerca de traps i siphons per l'enriquiment de la restricció d'abastabilitat.

D'altra banda fora una bona ampliació integrar a la interfície gràfica un editor de XdP per tal de facilitar a l'usuari la modelització de sistemes i l'anàlisi dinàmic dels mateixos. En aquest mateix camí, ampliar la definició basada en PNML per reconèixer xarxes i permetre carregar-les a l'editor segons els paràmetres de disseny especificats en aquest fitxer.

A més d'això, la interfície gràfica es podria fer més interactiva per tal que l'usuari pogués escollir les propietats sense haver d'escriure en format de text pla.

9 Bibliografia

1. **Murata, Tadao.** Petri Nets: Properties, Analysis and Applications. [aut. libro] IEEE. *Proceedings of the IEEE.* 1989, Vol. 77, 4, págs. 541-580.

2. **Desel, Jörg; Esparza, Javier;.** S-invariants and T-invariants. [aut. libro] Jörg Desel y Javier Esparza. *Free Choice Petri Nets.* s.l. : Cambridge University Press, 1995, 2.4, págs. 30-37.

3. **Desel, Jörg.** Basic linear algebraic techniques for place/transition nets. *Lectures on Petri Nets I: Basic Models.* s.l. : Springer Berlin / Heidelberg, 1998, págs. 257-308.

4. **Internautas, Comunidad de.** Wikipedia. [En línea] 2010. http://en.wikipedia.org/wiki/Linear_programming.

5. *Synthesis of Asynchronous Controllers using Integer Linear Programming.* **Carmona, Josep; Colom, José-Manuel; Cortadella, Jordi; Garcá-Vallés, Fernando.** [ed.] IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Vol. 25.

6. **Desel Jörg; Esparza, Javier;.** Stable predicates: siphons and traps. [aut. libro] Jörg Desel y Javier Esparza. *Free choice Petri nets.* s.l. : Cambridge University Press, 1995, 4.2, págs. 66-69.

7. lp_solve. [En línea] http://lpsolve.sourceforge.net/5.5/mps-format.htm.

8. **Desel, Jörg y Esparza, Javier.** *Free Choice Petri Nets.* s.l. : Cambridge Uniersity Press, 1995.

9.1 Altres fonts

- http://en.wikipedia.org
- http://docs.wxwidgets.org/stable/
- http://www.graphviz.org/Documentation.php
- http://www.gnu.org/software/gdb/

10 ANNEX 1: Teoria ampliada sobre XdP

L'estudi sobre XdP ha estat més ampli del que finalment s'ha aplicat a l'aplicació. A continuació es vol fer menció a tots aquells aspectes sobre XdP relacionats amb els que ja s'ha vist i que amplien el coneixement que es pot extreure de les xarxes a partir d'altres propietats i d'altres mètodes d'anàlisi.

Per seguir amb la mateixa estructura que s'ha usat a la part teòrica de la memòria, s'exposarà un conjunt de noves propietats i un conjunt de mètodes d'anàlisi que permetin restringir l'espai de solucions del problema de forma que els resultats obtinguts siguin vàlids al cent per cent.

10.1 Altres propietats

A més de les propietats ja mencionades anteriorment (veure apartat 2.1.2.2 d'aquest document) existeixen altres propietats comportamentals interessants que no s'ha inclòs en aquest projecte degut a que els mètodes d'anàlisi utilitzats no ens ho han permès.

10.1.1 Vivacitat

Definició

Una transició t d'una XdP marcada és **viva** si, per a cada marcatge M_i, existeix un marcatge M_j abastable des de M_i en el qual t es dispara.

Una XdP marcada és **viva** si cadascuna de les seves transicions està viva.

Una XdP és **morta** per un marcatge M_i si, per algun marcatge M_j successor de M_i , totes les transicions són mortes.

Una XdP és **parcialment viva** si conté transicions que sempre són vives i d'altres que no ho són.

Característiques

En una XdP viva es garanteix una operació lliure de punts morts independentment de la seqüència de dispars escollida.

La vivacitat és una propietat molt important ja que serveix per caracteritzar el **bloqueig** parcial o total d'un sistema.

Exemples

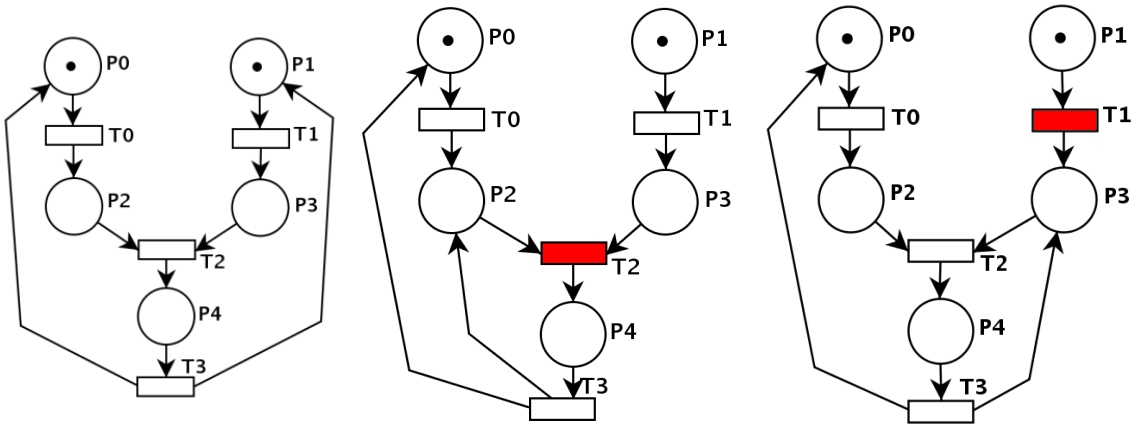


Figura 75: A l'esquerra, XdP viva; al mig, XdP morta (es bloqueja completament després de disparar {T0,T1},T2,T3,T0); a la dreta, XdP parcialment viva (T1 només es pot disparar una vegada).

10.1.2 Limitació

Definició

Un lloc p d'una XdP marcada és **k-limitat** per un marcatge M_0 si, per a qualsevol marcatge M_i successor de M_0, el nombre de marques a p és menor o igual a k, és a dir, sí i només sí existeix un k fix tal que $M_i(p) \leq k$ per a tot marcatge abastable des de M_0.

S'anomena **límit del lloc** p al menor enter k que verifica la desigualtat anterior.

Un lloc és **segur** si és 1-limitat.

Una XdP és **segura** o **binària** si cadascun dels seus llocs és segur.

Característiques

La k-limitació d'una xarxa té el seu interès en què garanteix la **finitud** dels seus marcatges abastables. Des d'un punt de vista pràctic, una xarxa k-limitada es pot implementar amb un conjunt de recursos finit.

És útil per a XdP que modelen sistemes amb memòries intermèdies o registres ja que verificant que la xarxa és limitada o segura es garanteix que no hi haurà desbordaments.

Exemples

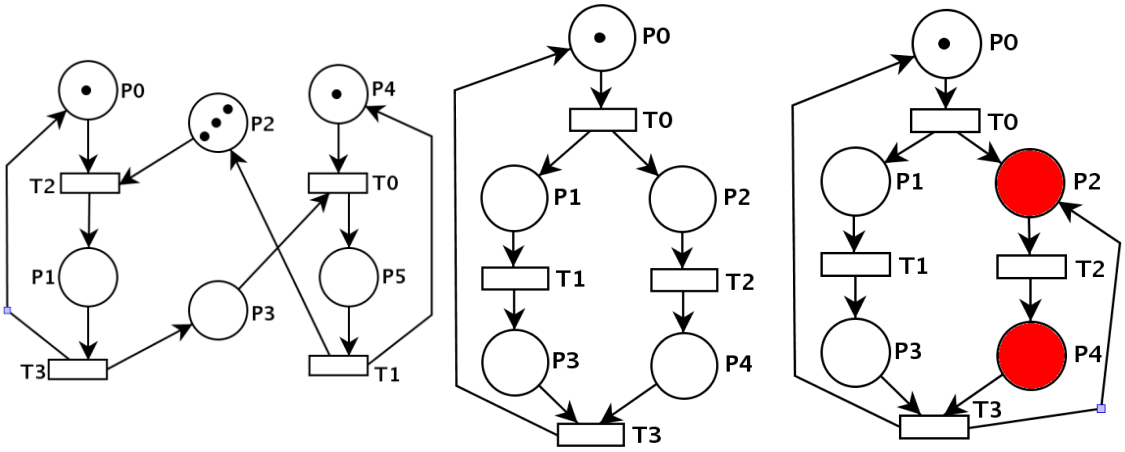


Figura 76: A l'esquerra, XdP 3-limitada ; al mig, XdP binària ; a la dreta, XdP no limitada.

10.1.3 Conservació

Definició

Una XdP amb un marcatge inicial és M_0 és **conservativa** si, per a qualsevol marcatge M_i abastable des de M_0, el nombre total de marques a la xarxa és el mateix, és a dir, si es verifica per a tot marcatge que $\sum_{p_i \in P} M(p_i) = \sum_{p_i \in P} M_0(p_i)$.

Una XdP amb un marcatge inicial M_0 és **conservativa respecte d'un vector de pes** w ($w=(w1, w2, \ldots wn)$, on $wi \geq 0$) si, per a qualsevol marcatge M abastable des de M_0, la suma del producte del nombre de marques de cada lloc pel factor de pes corresponent és constant, és a dir, si per a tot marcatge $M \in R(M_0)$ es verifica que $\sum_{i=1}^{n} w_i \cdot M(p_i) = \sum_{i=1}^{n} w_i \cdot M_0(p_i)$.

Característiques

El concepte de conservació està relacionat amb el nombre de recursos disponible, que no pot variar durant l'execució de la XdP.

Cal tenir en compte que la conservació estricta és una relació molt forta i que la majoria de vegades es parla de la conservació respecte d'un vector de pesos.

Exemples

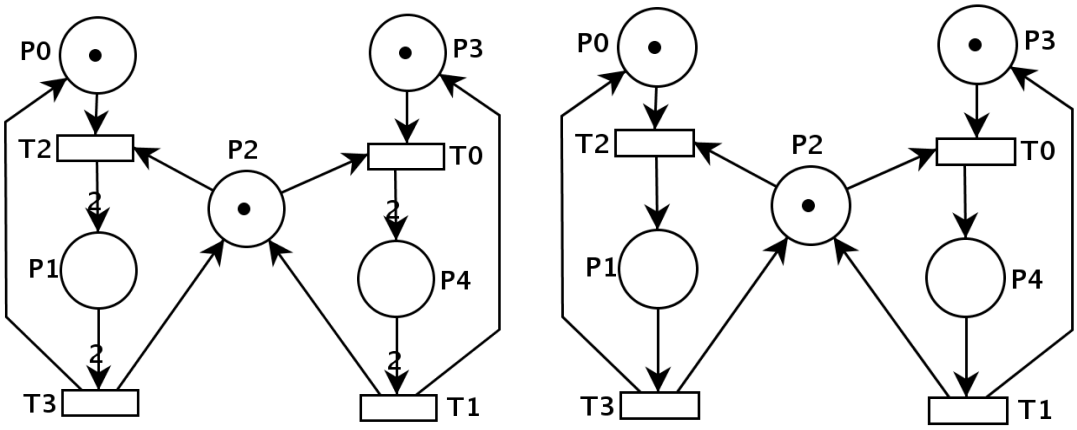


Figura 77: A l'esquerra, XdP conservativa estricta; a la dreta, XdP conservativa respecte del vector de pes w=(1,1,1,2,2).

10.2 Altres mètodes d'anàlisi

Tal i com s'ha explicat anteriorment, s'ha aplicat tècniques estructurals per a l'anàlisi de les XdP, concretament s'ha utilitzat l'equació de marcatge i la teoria dels invariants. Però existeixen altres mètodes, tant dins de tècniques estructurals com en d'altres. A continuació es fa una ampliació de les tècniques enumeratives i les estructurals, donant exemples d'aplicació d'aquestes tècniques a l'anàlisi de XdP, i es fa una introducció a les tècniques de transformació, tot amb la finalitat d'obrir els horitzons de les funcionalitats de l'aplicació en el futur.

10.2.1 Tècniques enumeratives: aplicacions

A partir de l'arbre d'abastabilitat es pot verificar propietats com per exemple:

- Bloqueig del sistema.
- Existència de parts del sistema que mai entren en funcionament.
- Existència de llocs no segurs.
- Existència d'estats absorbents (dead-locks).
- Existència de cicles sense fi.
- Finitud de l'espai d'estats.
- Absència de conflictes.
- Verificació d'especificacions.

A continuació s'especifica un conjunt de propietats que es poden verificar a partir de l'arbre o graf d'abastabilitat. Es parlarà, de forma genèrica, de l'arbre d'abastabilitat.

Limitació

A partir de l'arbre d'abastabilitat es pot establir que:

- Un lloc p_i és **limitat** si i només si l'arbre d'abastabilitat no conté cap marcatge on $m_i=w$.
- Una XdP és **limitada** si a l'arbre d'abastabilitat no apareix cap w.
- Una XdP és **segura** (**binària**) si als marcatges de l'arbre d'abastabilitat només apareixen 0s i 1s.
- Si M és abastable des de M_0, llavors existeix un marcatge M' tal que $M \leq M'$.

Conservació

Es pot verificar si una XdP és conservativa a partir de l'arbre d'abastabilitat:

- Si aquest conté alguna ω, la XdP només pot ser **conservativa** si el pes corresponent al lloc on apareix ω és 0.
- Si no existeix cap ω, el sistema té un nombre finit d'estats i la conservació es pot verificar directament sobre l'arbre.

Vivacitat

Aquesta propietat és difícil de comprovar en XdP en què el seu arbre d'abastabilitat conté ω.

A continuació es mostra dos XdP, una viva i l'altra morta, ambdues amb el mateix arbre d'abastabilitat:

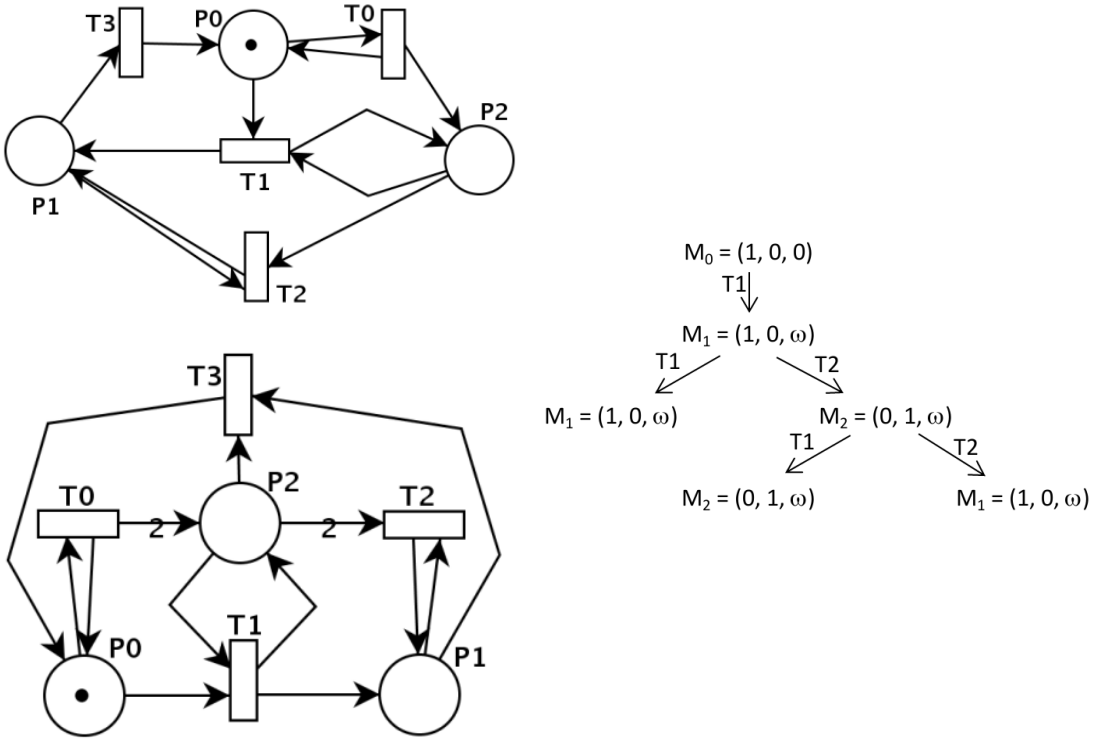


Figura 78: a l'esquerra dues XdP, a dalt, viva, a baix, morta. A la dreta l'arbre d'abastabilitat d'ambdues

10.2.2 Tècniques estructurals

10.2.2.1 Ús d'invariants per la verificació de propietats comportamentals

S-invariants i les propietats comportamentals

A continuació es mostra diferents teoremes sobre l'aplicació dels S-invariants per verificar les propietats comportamentals de les XdP:

- **Condició necessària per la vivacitat**

  Una XdP *N* és un sistema viu per un marcatge inicial M_0 si i només si cada S-invariant semi-positiu *I* satisfà $I \cdot M_0 > 0$.

- **Condició suficient per la limitació**

  Si una XdP *N* té un S-invariant positiu per un marcatge inicial M_0, llavors la xarxa està limitada.

- **Condició necessària per l'abastabilitat**

  Es diu que dos marcatges *M* i *L* d'una XdP *N* **concorden** per tots els S-invariants si $I \cdot M = I \cdot L$ per a tot S-invariant *I* de la xarxa. També es pot dir que *M* i *L* concorden si l'equació $M + N \cdot X = L$ té alguna solució racional per *X*.

  Donats una XdP *N* i un marcatge inicial M_0, *M* és un marcatge abastable des de M_0 si *M* iM_0 concorden per a tots els seus S-invariants.

T-invariants i les propietats comportamentals

Una XdP *N* és **ben formada** si existeix un marcatge M_0 tal que el sistema és viu i limitat. Per la definició del sistema, les xarxes ben formades són connexes i tenen al menys una transició i un place.

Qualsevol sistema viu i limitat representat per una XdP *N* i un marcatge inicial M_0 té un marcatge abastable *M* i una seqüència de dispars $M \xrightarrow{\sigma} M$ tal que totes les transicions de N es disparen a σ^2. D'aquesta afirmació es pot extreure que qualsevol xarxa ben formada té un T-invariant, ja que el vector de parik que representa la seqüència de dispars $M \xrightarrow{\sigma} M$ és un t-invariant positiu.

S-invariants, T-invariants i les propietats comportamentals

Qualsevol xarxa connexa amb un S-invariant i un T-invariant positius és fortament connexa.

10.2.2.2 Predicats Estables

La informació d'aquest punt s'extreu de l'apartat de "Stable predicates: siphons and traps" (5) del llibre ja mencionat "*Free Choice Petri Nets*" i de l'article "*Basic Linear Algebraic Techniques for Place/Transition Nets*", de Jörg Desel (3).

Definim $\mathcal{M}$ com el conjunt de marcatges *M* d'una XdP *N* tals que el sistema *(N, M)* és viu. De la definició de vivacitat es pot inferir que si *(N, M)* és viu i el marcatge *L* és abastable des de *M*, llavors el sistema *(N, L)* també és viu. Per tant, tots els marcatges abastables a partir d'un marcatge de $\mathcal{M}$ estan també a $\mathcal{M}$, propietat que s'anomena **estabilitat**. Un predicat *L* és estable si es compleix que $(M \in \mathcal{M} \wedge M \xrightarrow{t} L) \Rightarrow L \in \mathcal{M}$.

Als dos següents apartats es parla de predicats estables que es poden derivar directament de l'estructura de la xarxa.

10.2.2.2.1 Traps

Un **trap** és un conjunt de llocs P tals que $\cdot P \subset P \cdot$, és a dir, per tota transició es compleix $|\cdot t| \times |A \cap t \cdot| \geq |A \cap \cdot t|$ (si la transició consumeix al menys una marca d'un lloc del trap, també deixa, com a mínim, una marca a un lloc del trap).

Els traps generen condicions necessàries d'abastabilitat: considerem dos marcatges, M_0 i *M*, d'una XdP *N* tals que *M* és abastable des de M_0. Llavors, tots els traps de *N* que contenen un lloc marcat a M_0 també contenen un lloc marcat a *M*.

D'aquesta teorema es pot extreure un **test de no-abastabilitat** per un marcatge *M*:

1. Escollir un conjunt de llocs.
2. Comprovar la propietat de trap.
3. Verificar que el marcatge inicial marca al menys un lloc del trap mentre que el marcatge *M* no en marca cap.

Existeix un trap per la xarxa *N* que

- conté un lloc *p* que satisfà $M_0(p) \geq 1$ i
- no conté cap lloc *p* que satisfà $M(p) \geq 1$,

si i només si existeix solució pel següent sistema d'inequacions:

$M_0 \cdot x > 0$, $\quad [\exists p: M_0(p) \geq 1]$

$M \cdot x \leq 0$, $\quad [\nexists p: M(p) \geq 1]$

$(|\cdot t| \cdot \mathcal{X}(t\cdot) - \mathcal{X}(\cdot t)) \geq 0$ per tota transició *t*, on X(·t) és el vector característic del pre-conjunt de t i X(t·) és el del seu post-conjunt.

$x \geq 0$.

Es pot restringir el sistema per obtenir només solucions binàries, *x* és a {0, 1}* substituint la segona inequació per $M \cdot x = 0$.

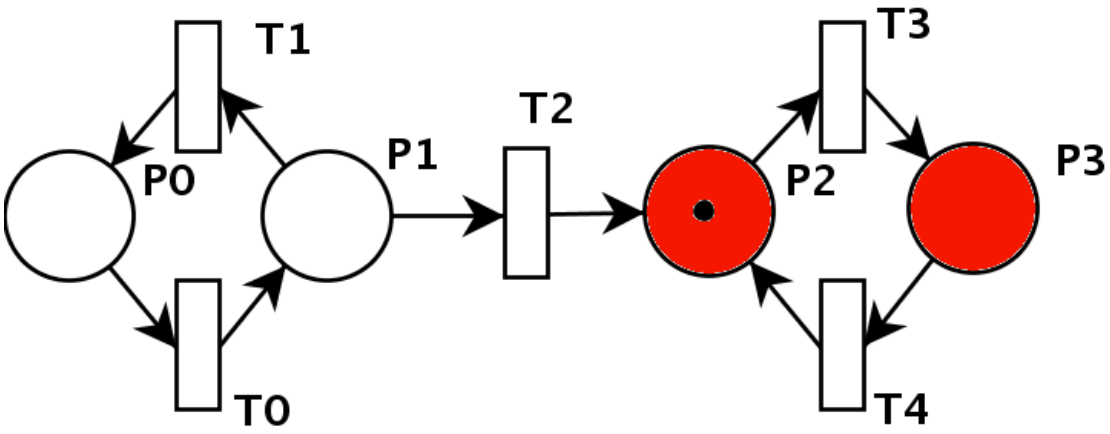


Figura 79: El conjunt {P2, P3} és un trap del model

10.2.2.2.2 Siphons

Un **siphon** és un conjunt de llocs P tals que $R \cdot \subseteq \cdot R$ (un siphon que no conté cap marcatge inicial no podrà ser marcat per cap seqüència de dispars).

Els siphons generen condicions necessàries d'abastabilitat: considerem dos marcatges, M_0 i M, d'una XdP N tals que M és abastable des de M_0. Llavors, tots els siphons de N que no contenen cap lloc marcat a M_0 tampoc contenen cap lloc marcat a M.

D'aquesta teorema es pot extreure un **test de no-abastabilitat** per un marcatge M:

1. Escollir un conjunt de llocs.
2. Comprovar la propietat de siphon.
3. Verificar que el marcatge inicial no marca cap lloc del siphon mentre que el marcatge M en marca com a mínim un.

Existeix un siphon per la xarxa N que

- no conté cap lloc p que satisfà $M_0(p) \geq 1$ i
- conté algun lloc p que satisfà $M(p) \geq 1$,

si i només si existeix solució pel següent sistema d'inequacions:

$M_0 \cdot x \leq 0$, $[\nexists p: M_0(p) \geq 1]$

$M \cdot x > 0$, $[\exists p: M(p) \geq 1]$

$(|t\cdot| \cdot \mathcal{X}(\cdot t) - \mathcal{X}(t\cdot)) \geq 0$ per tota transició t,

$x \geq 0$.

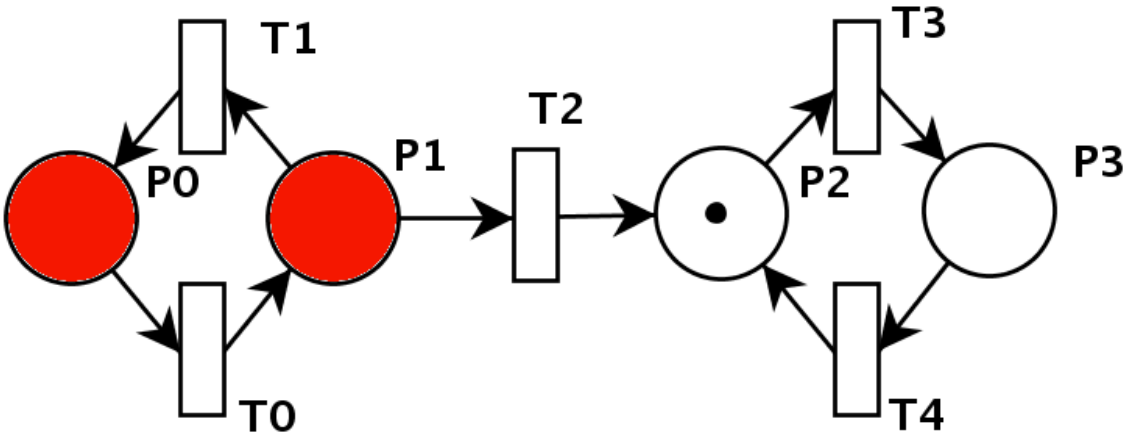


Figura 80: El conjunt {P0, P1} és un siphon del model

10.2.3 Introducció a les tècniques de transformació

10.2.3.1 Tècniques de transformació

L'objectiu d'aquest conjunt de tècniques és reduir la mida dels models mitjançant regles de reducció que preserven les propietats que es vol estudiar.

A continuació es mostra uns quants exemples de reducció que preserven la vivacitat i la k-limitació:

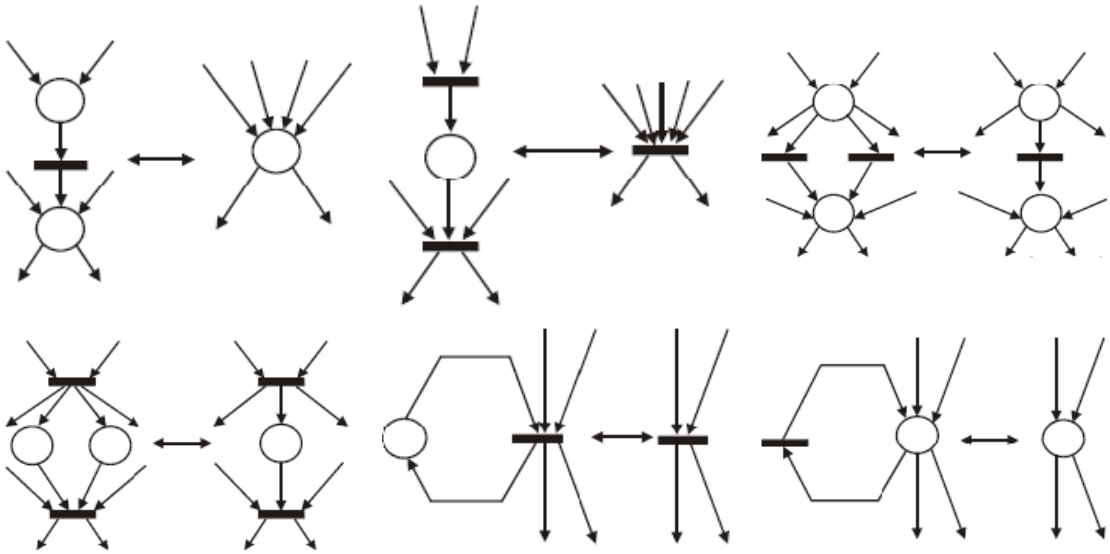


Figura 81: Conjunt de tècniques amb el que és possible reduir la complexitat de càlcul de la vivacitat i la limitació d'un sistema.

10.3 Subclasses de XdP

Podem distingir les següents subclasses de XdP:

- **Graf d'Estats (GE)** o **Màquina d'Estats (ME)**:
  - Cada transició té exactament una entrada i una sortida:
    $|{}^\bullet t| = |t^\bullet| = 1, \forall\, t \in T$.
  - Característiques:
    - És estrictament conservativa.
    - Pot representar conflictes.
    - No pot representar paral·lelisme, concurrència ni sincronització
- **Graf marcat (GM)** o Graf de sincronització:
  - Cada lloc té exactament una entrada i una sortida:
    $|{}^\bullet p| = |p^\bullet| = 1, \forall\, p \in P$.
  - Característiques:
    - Pot representar paral·lelisme, concurrència i sincronització.
    - No pot representar conflicte.
- **Xarxa de lliure elecció (XLE)**:
  - Cada lloc p és:
    - o bé l'únic lloc del pre-conjunt d'una transició,
    - o bé hi ha com a màxim una transició que té p al seu pre-conjunt
      $\forall\, p_1, p_2 \in P,\ p_1{}^\bullet \cap p_2{}^\bullet \neq \emptyset \Rightarrow |p_1{}^\bullet| = |p_2{}^\bullet| = 1$, equivalentment
      $\forall\, p \in P,\ |p^\bullet| \leq 1$ *or* ${}^\bullet(p^\bullet) = \{p\}$.
  - Característiques:
    - Pot representar concurrència i conflicte, tot i que d'una manera més restringida que el model general: si un lloc és entrada de més d'una transició (conflicte potencial), és l'única entrada d'aquestes transicions, per tant, o bé totes les transicions conflictives estan simultàniament actives o cap d'elles ho està. Això permet la lliure elecció (resolució del conflicte) de la transició que es dispara ja que la presència de marques en altres llocs no intervé en l'elecció de la transició que es dispara.
      Aquesta forma restringida de conflicte permet trobar condicions necessàries i suficients per definir la XdP com a viva i segura.
- **Xarxa d'Elecció Asimètrica (XEA) o Xarxa de Petri simple (XS)**:
  - Cada transició té com a màxim un lloc d'entrada compartit amb una altra transició:
    $\forall\, p_1, p_2 \in P,\ p_1{}^\bullet \cap p_2{}^\bullet \neq \emptyset \Rightarrow p_1{}^\bullet \subseteq p_2{}^\bullet$ *or* $p_1{}^\bullet \supseteq p_2{}^\bullet$.
  - Característiques:
    - La concurrència i el conflicte (és a dir, la confusió), poden ocórrer, però no de forma asimètrica.