Soutenance de thèse

Schémas de formules et de preuves en logique propositionnelle

Vincent Aravantinos

Nicolas	Peltier	Directeur
Ricardo	Caferra	Co-directeur

Jury :

Maria Paola	Bonacina	Rapporteure
Dominique	Duval	Examinatrice
Miki	Hermann	Rapporteur
Michel	Parigot	Examinateur
Nicolas	Peltier	Directeur
Brigitte	Plateau	Présidente

23 septembre 2010

Plan

Introduction générale : la déduction automatique

Introduction à la thèse : les schémas de formules

Déduction automatique et schémas de formules

Analyse d'un point technique : décidabilité des schémas réguliers

Conclusion

Plan

Introduction générale : la déduction automatique

Introduction à la thèse : les schémas de formules

Déduction automatique et schémas de formules

Analyse d'un point technique : décidabilité des schémas réguliers

Conclusion

Déduction automatique

« déduction automatique »

$\approx$

utilisation de l'ordinateur pour le raisonnement

Dans le cadre de cette thèse :

conception d'un algorithme qui prend en entrée une formule d'une logique et prouve sa validité

Exemple : la logique propositionnelle

Syntaxe :

formules composées de $\begin{cases} \text{« et » (noté } \wedge\text{),} \\ \text{« ou » (noté } \vee\text{),} \\ \text{« non » (noté } \neg\text{)} \\ \text{« variables (propositionnelles) »} \end{cases}$

p.ex. $P \vee \neg P$, $P \wedge \neg P$, $(P \wedge Q) \vee (\neg P \wedge R)$

Sémantique :

Variable interprétée soit par vrai, soit par faux.

Formule $= \begin{cases} \textbf{valide} \text{ ssi vraie pour toute interprétation.} \\ \textbf{satisfaisable} \text{ ssi vraie dans une interprétation.} \\ \textbf{insatisfaisable} \text{ ssi fausse pour toute interprétation.} \end{cases}$

p.ex.

$P \vee \neg P$	valide
$P \wedge \neg P$	insatisfaisable
$(P \wedge Q) \vee (\neg P \wedge R)$	satisfaisable mais non valide

$\Rightarrow$ Il existe un algorithme capable de dire si une formule propositionnelle est satisfaisable ou non.

Problématique

Logique + expressive ⇒ déduction automatique + difficile

Exemple :

	logique propositionnelle	*NP-dur*
	μ-calcul	*EXPTIME-dur*
	logique du 1er ordre	*semi-décidable*
	logiques d'ordre supérieur	*non semi-décidables*

expressivité ↓ ↓ difficulté

Rappel : « indécidable » = infaisable par une machine (de Turing)

But (au sens large) de cette thèse : Exprimer plus de formules, tout en préservant de bonnes propriétés calculatoires.

À quoi peut servir la déduction automatique ?

- application usuelle : **vérification de systèmes (logiciel, circuit, etc.)**
  - façon simple de vérifier qu'un système fonctionne bien : *tests*
  - problème : on ne peut pas tout tester
    $\Rightarrow$ bugs potentiels
  - solution :
    démontrer *formellement* qu'un système fonctionne bien
  - nouveau problème : *beaucoup* de théorèmes à démontrer
  - (une) solution : démontrer tous ces théorèmes avec une machine *grâce à la déduction automatique*
- formalisation des mathématiques
- enseignement des mathématiques
- (indirectement) web sémantique

Plan

Introduction générale : la déduction automatique

Introduction à la thèse : les schémas de formules

Déduction automatique et schémas de formules

Analyse d'un point technique : décidabilité des schémas réguliers

Conclusion

Schémas

Notion de schéma fréquemment rencontrée :
« représentation finie d'un ensemble infini d'objets structurellement similaires »

- en mathématiques (somme itérée) :

$$\sum_{i=1}^{n} x_i$$

$$\forall n \geq 1, \forall x_1, \ldots, x_n \in \mathbb{R} : \left(\sum_{i=1}^{n} x_i^2\right) \leq \left(\sum_{i=1}^{n} x_i\right)^2$$

- en logique (règles d'inférence, schémas d'axiomes) :

$$\frac{A \qquad A \Rightarrow B}{B}$$

$$\forall x(A(0) \wedge \forall x(A(x) \Rightarrow A(s(x)))) \Rightarrow \forall x(A(x))$$

- en déduction automatique (schémas itérés de termes) :

$$\mathrm{f}^n(\mathrm{a})$$

Définition informelle

En première approximation :

« schémas itérés propositionnels »

=

formules propositionnelles
+ $\bigwedge_{i=1}^{n}$ (« ET pour i variant de 1 à n »)
+ $\bigvee_{i=1}^{n}$ (« OU pour i variant de 1 à n »)
où n est une variable du langage

Exemple :

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

Signification :
Un schéma = une **infinité** de formules propositionnelles.

Définition formelle

Syntaxe :

$$\phi ::= \phi_1 \vee \phi_2 \,|\, \phi_1 \wedge \phi_2 \,|\, \bigwedge_{i=1}^{n-k} \phi \,|\, \bigvee_{i=1}^{n-k} \phi \,|\, p_{expr} \,|\, \neg p_{expr}$$

où :

- $k \in \mathbb{Z}$
- i, n variables entières
- *expr* expression arithmétique linéaire quelconque

Exemple :

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

Sémantique : Un schéma est insatisfaisable ssi chaque formule propositionnelle obtenue en donnant une valeur à n est insatisfaisable.

$$\begin{tikzcd} \bigwedge_{i=1}^{n} \phi \arrow[r, bend left, "n=1"] & \phi[1/i] \arrow[d] \\ & insat \end{tikzcd}$$

Définition formelle

Syntaxe :

$$\phi ::= \phi_1 \vee \phi_2 \,|\, \phi_1 \wedge \phi_2 \,|\, \bigwedge_{i=1}^{n-k} \phi \,|\, \bigvee_{i=1}^{n-k} \phi \,|\, p_{expr} \,|\, \neg p_{expr}$$

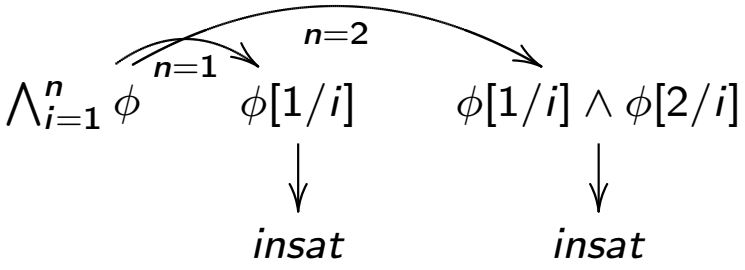
où :

- $k \in \mathbb{Z}$
- i, n variables entières
- *expr* expression arithmétique linéaire quelconque

Exemple :

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

Sémantique : Un schéma est insatisfaisable ssi chaque formule propositionnelle obtenue en donnant une valeur à n est insatisfaisable.



Définition formelle

Syntaxe :

$$\phi ::= \phi_1 \vee \phi_2 \,|\, \phi_1 \wedge \phi_2 \,|\, \bigwedge_{i=1}^{n-k} \phi \,|\, \bigvee_{i=1}^{n-k} \phi \,|\, p_{expr} \,|\, \neg p_{expr}$$

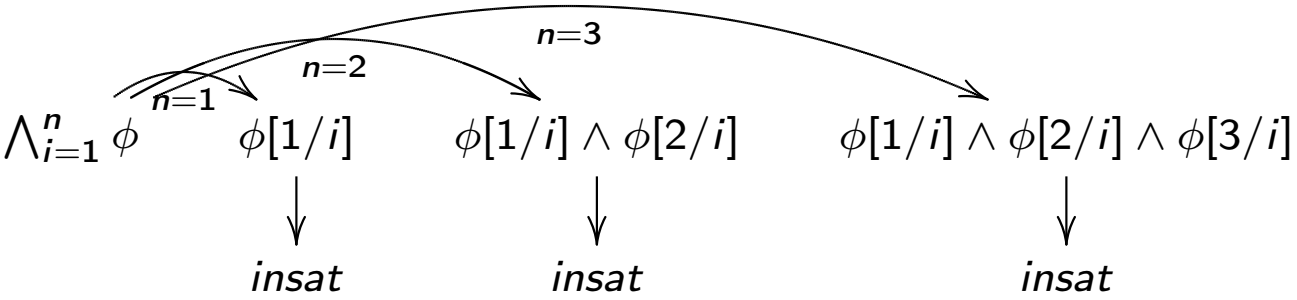
où :

- $k \in \mathbb{Z}$
- i, n variables entières
- *expr* expression arithmétique linéaire quelconque

Exemple :

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

Sémantique : Un schéma est insatisfaisable ssi chaque formule propositionnelle obtenue en donnant une valeur à n est insatisfaisable.



Définition formelle

Syntaxe :

$$\phi ::= \phi_1 \vee \phi_2 \,|\, \phi_1 \wedge \phi_2 \,|\, \bigwedge_{i=1}^{n-k} \phi \,|\, \bigvee_{i=1}^{n-k} \phi \,|\, p_{expr} \,|\, \neg p_{expr}$$

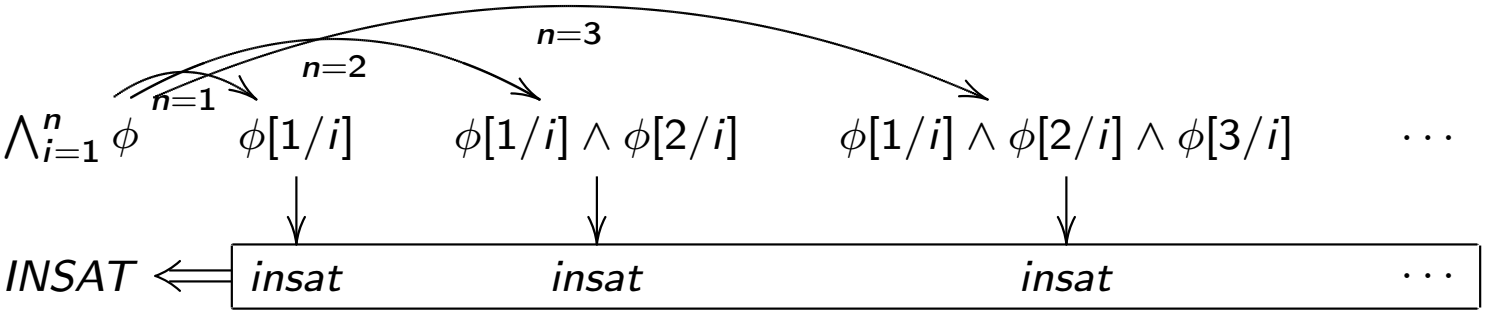
où :

- $k \in \mathbb{Z}$
- i, n variables entières
- *expr* expression arithmétique linéaire quelconque

Exemple :

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

Sémantique : Un schéma est insatisfaisable ssi chaque formule propositionnelle obtenue en donnant une valeur à n est insatisfaisable.



Exemple précédent

$$p_1 \wedge \bigwedge_{i=1}^{n}(p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1} \xrightarrow{n=0} p_1 \wedge \neg p_1 \rightarrow \text{insat}$$

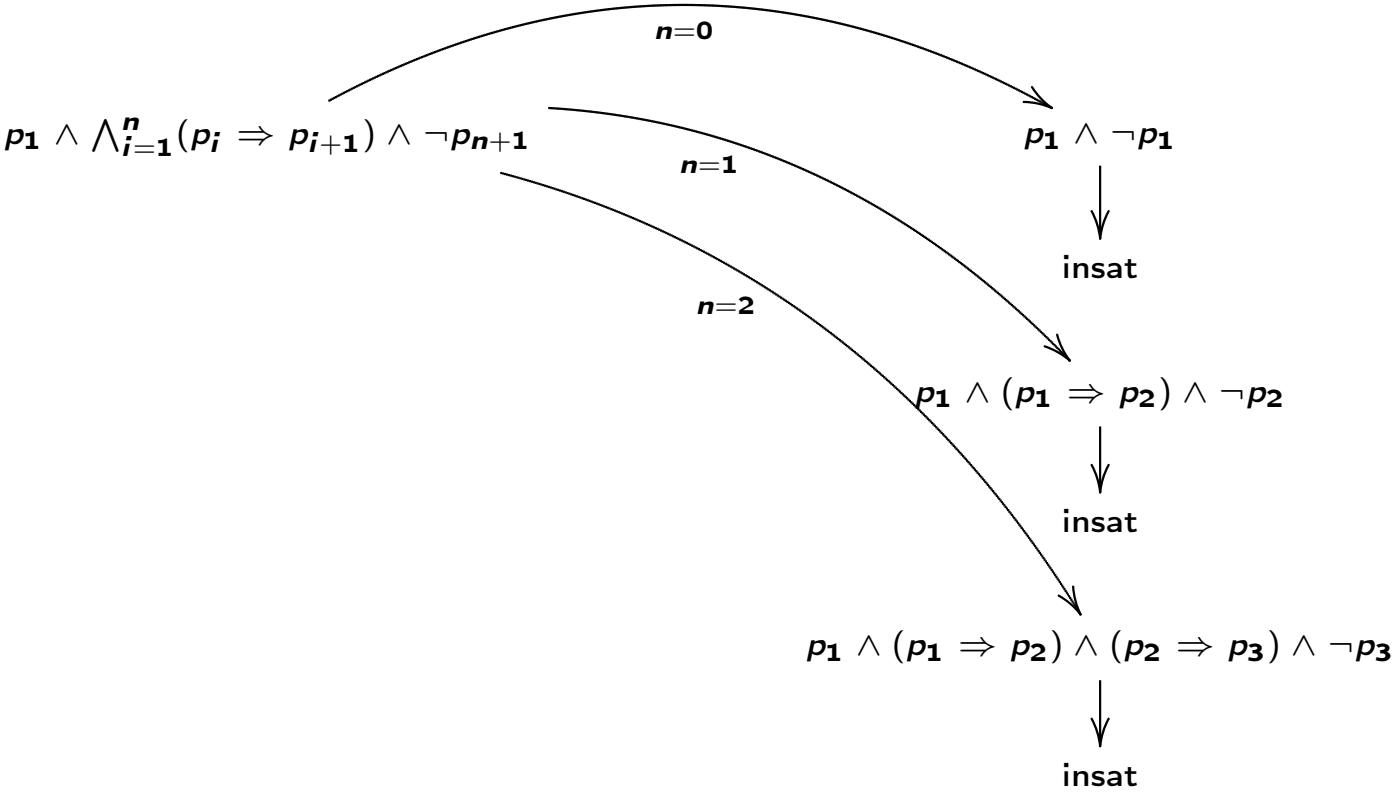
Exemple précédent

$$p_1 \wedge \bigwedge_{i=1}^{n}(p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

$n=0$: $p_1 \wedge \neg p_1 \rightarrow$ insat

$n=1$: $p_1 \wedge (p_1 \Rightarrow p_2) \wedge \neg p_2 \rightarrow$ insat

Exemple précédent



Exemple précédent

$$p_1 \wedge \bigwedge_{i=1}^{n}(p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

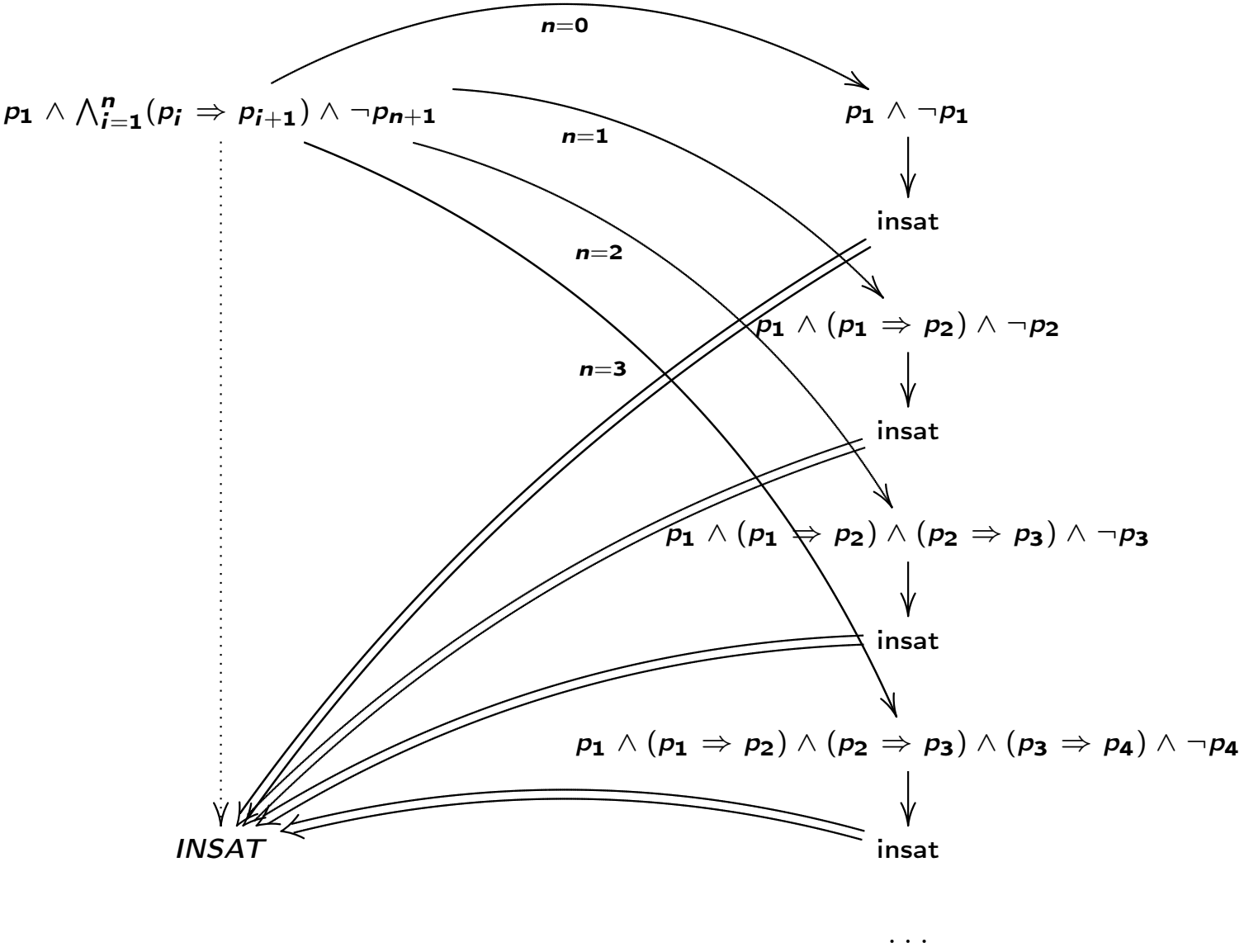
$n=0$: $p_1 \wedge \neg p_1 \rightarrow$ insat

$n=1$: $p_1 \wedge (p_1 \Rightarrow p_2) \wedge \neg p_2 \rightarrow$ insat

$n=2$: $p_1 \wedge (p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_3) \wedge \neg p_3 \rightarrow$ insat

$n=3$: $p_1 \wedge (p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_3) \wedge (p_3 \Rightarrow p_4) \wedge \neg p_4 \rightarrow$ insat

Exemple précédent



Que peut-on exprimer avec les schémas de formules ?

- exemple concret : vérification de circuits
  $\longrightarrow$ raisonnement indépendant du nombre de bits
- problèmes de satisfaction de contrainte (p.ex. les n-reines)
- assertions sur des *classes* de graphes (p.ex. la coloration)
- assertions sur des automates finis (p.ex. l'inclusion)
- motivation importante : formalisation des mathématiques

  p.ex. avec l'outil d'élimination des coupures CERES

  (projet ASAP commun avec l'équipe d'Alex Leitsch à Vienne)

  $\longrightarrow$ schémas au *premier ordre*

  (non abordé dans la thèse, cf travaux futurs)

Travaux connexes

- **logiques de points fixes ($\approx \mu$-calcul au 1er ordre) :**

  p.ex. $\bigwedge_{i=1}^{n} p_i \rightsquigarrow \mu X(n).(\exists m.n = \mathrm{succ}(m) \Rightarrow p_m \wedge X(m)) \wedge (n = 0 \Rightarrow \mathrm{V})$

  (mais peu de classes décidables connues)

- **logique du premier ordre, en transformant les itérations en quantificateurs bornés :**

  $$\text{p.ex. } \bigwedge_{i=1}^{n} p_i \rightsquigarrow \forall n.\forall i.1 \leq i \leq n \Rightarrow p(i)$$

  (mais perte au niveau de la sémantique)

- **logique du premier ordre avec conséquence inductive, en transformant les itérations en fonctions récursives :**

  $$\text{p.ex. } \bigwedge_{i=1}^{n} p_i \rightsquigarrow \forall n.f(n) = \tilde{\mathrm{F}} \text{ où } \begin{cases} f(0) = \tilde{\mathrm{V}} \\ f(i+1) = p(i) \,\tilde{\wedge}\, f(i) \\ \tilde{\wedge}\,,\ \tilde{\mathrm{V}} \text{ et } \tilde{\mathrm{F}} \text{ sont des symboles de fonction} \end{cases}$$

  (mais peu de classes décidables connues)

Plan

Introduction générale : la déduction automatique

Introduction à la thèse : les schémas de formules

Déduction automatique et schémas de formules

Analyse d'un point technique : décidabilité des schémas réguliers

Conclusion

Semi-décidabilité

Proposition :

*La **satisfaisabilité** des schémas est semi-décidable.*

c.-à-d. si un schéma a un modèle, alors on trouvera ce modèle.

Exemple :

$$p_n \wedge \neg p_0 \wedge \neg p_1 \xrightarrow{n=0} p_0 \wedge \neg p_0 \wedge \neg p_1 \downarrow \mathit{insat}$$

Théorème :

*L'**insatisfaisabilité** des schémas est indécidable.*
L'insatisfaisabilité des schémas monadiques est indécidable.
L'insatisfaisabilité des schémas plats est indécidable.

$\longrightarrow$ situation duale de la logique du premier ordre

(mais similaire à la théorie des modèles finis)

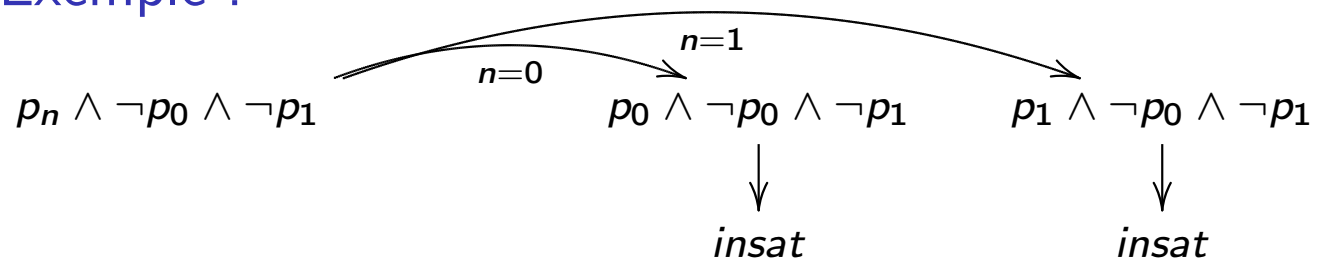
Semi-décidabilité

Proposition :

*La **satisfaisabilité** des schémas est semi-décidable.*

c.-à-d. si un schéma a un modèle, alors on trouvera ce modèle.

Exemple :



Théorème :

*L'**insatisfaisabilité** des schémas est indécidable.*

L'insatisfaisabilité des schémas monadiques est indécidable.

L'insatisfaisabilité des schémas plats est indécidable.

$\longrightarrow$ situation duale de la logique du premier ordre

(mais similaire à la théorie des modèles finis)

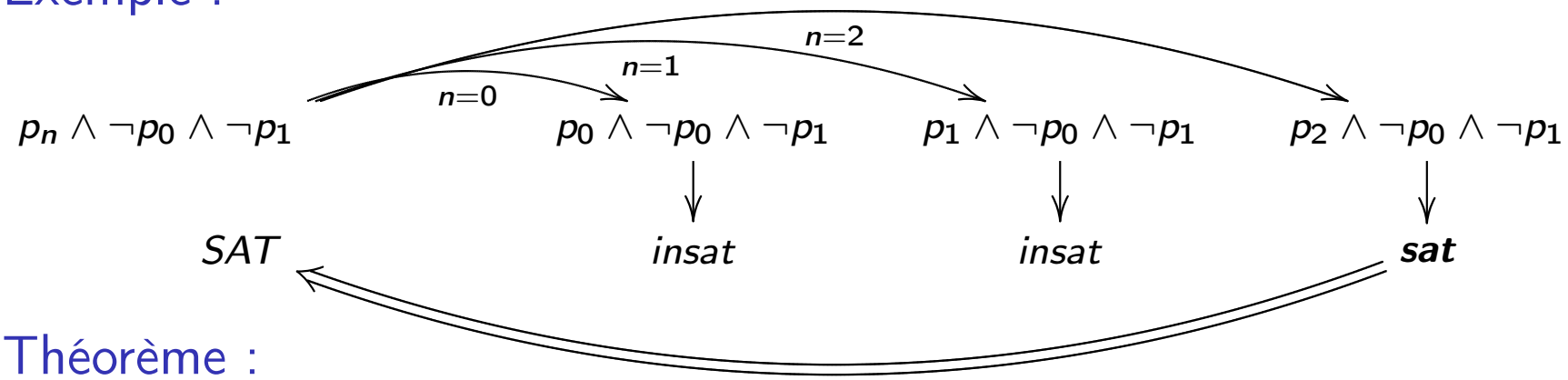
Semi-décidabilité

Proposition :

*La **satisfaisabilité** des schémas est semi-décidable.*

c.-à-d. si un schéma a un modèle, alors on trouvera ce modèle.

Exemple :



Théorème :

*L'**insatisfaisabilité** des schémas est indécidable.*

L'insatisfaisabilité des schémas monadiques est indécidable.

L'insatisfaisabilité des schémas plats est indécidable.

$\longrightarrow$ situation duale de la logique du premier ordre

(mais similaire à la théorie des modèles finis)

Déduction auto en logique propositionnelle : DPLL (1/2)

Rappel : procédure de Davis-Putnam-Logemann-Loveland (DPLL)
$\longrightarrow$ procédure la plus utilisée par les SAT-solvers

$P \wedge (P \Rightarrow Q) \wedge \neg Q$ est-elle insatisfaisable ?

$$P \wedge (P \Rightarrow Q) \wedge \neg Q$$

si $P = \text{vrai}$ $\qquad$ si $P = \text{faux}$

$$\mathrm{V} \wedge (\mathrm{V} \Rightarrow Q) \wedge \neg Q = Q \wedge \neg Q \qquad \mathrm{F} \wedge (\mathrm{F} \Rightarrow Q) \wedge \neg Q = \boxed{\mathrm{F}}$$

si $Q = \text{vrai}$ $\qquad$ si $Q = \text{faux}$

$$\mathrm{V} \wedge \neg \mathrm{V} = \boxed{\mathrm{F}} \qquad \mathrm{F} \wedge \neg \mathrm{F} = \boxed{\mathrm{F}}$$

Dans tous les cas, on a $\mathrm{F} \Rightarrow$ *formule insatisfaisable*

Note : À la fin : soit F, soit V.

Déduction auto en logique propositionnelle : DPLL (2/2)

Présentation de DPLL sous forme de tableaux :

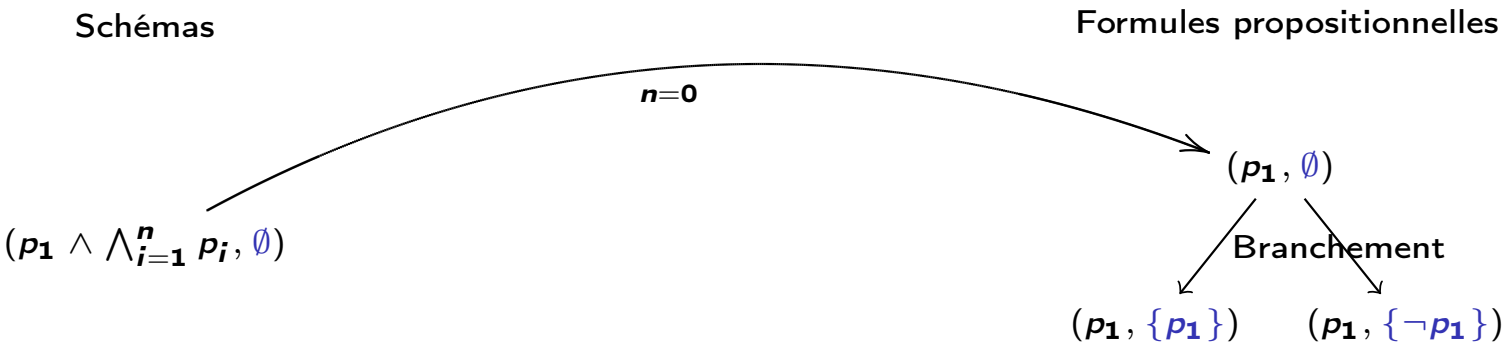
- construction d'un arbre (« tableau ») :
  - nœuds étiquetés par des paires (formule, ensemble de littéraux)
  - racine étiquetée par la formule à réfuter et l'ensemble vide
  - fils obtenus par application de règles :

Branchement	Évaluation (pos.)	Évaluation (neg.)
(ϕ, L) → $(\phi, L \cup \{X\})$, $(\phi, L \cup \{\neg X\})$	$(\phi[X], L \cup \{X\}) \downarrow (\phi[\mathrm{V}], L \cup \{X\})$	$(\phi[X], L \cup \{\neg X\}) \downarrow (\phi[\mathrm{F}], L \cup \{\neg X\})$
pour une variable prop. X de ϕ		

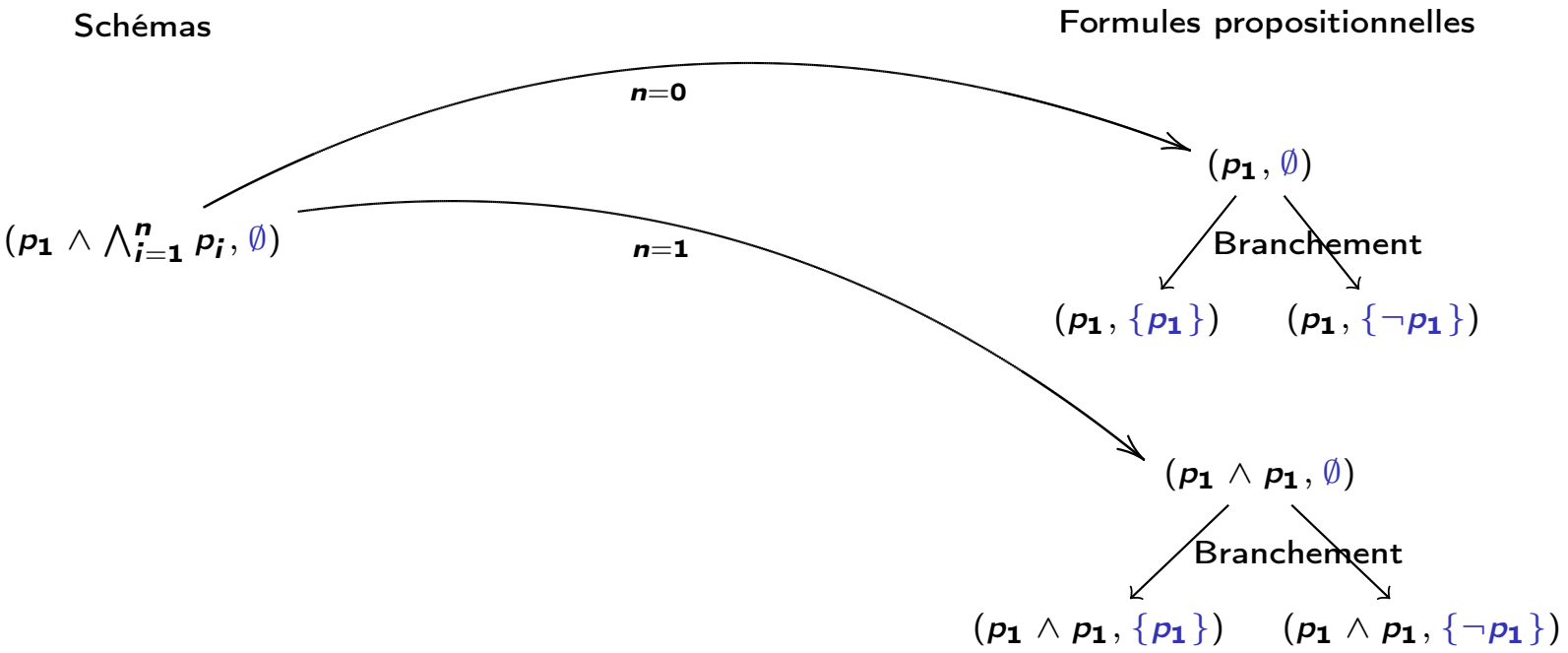
+ simplifications algébriques (V élément neutre de $\wedge$, etc.)

- à la fin de la construction :
  Si toutes les feuilles sont étiquetées par F (et un ensemble quelconque de littéraux) alors la formule est insatisfaisable.

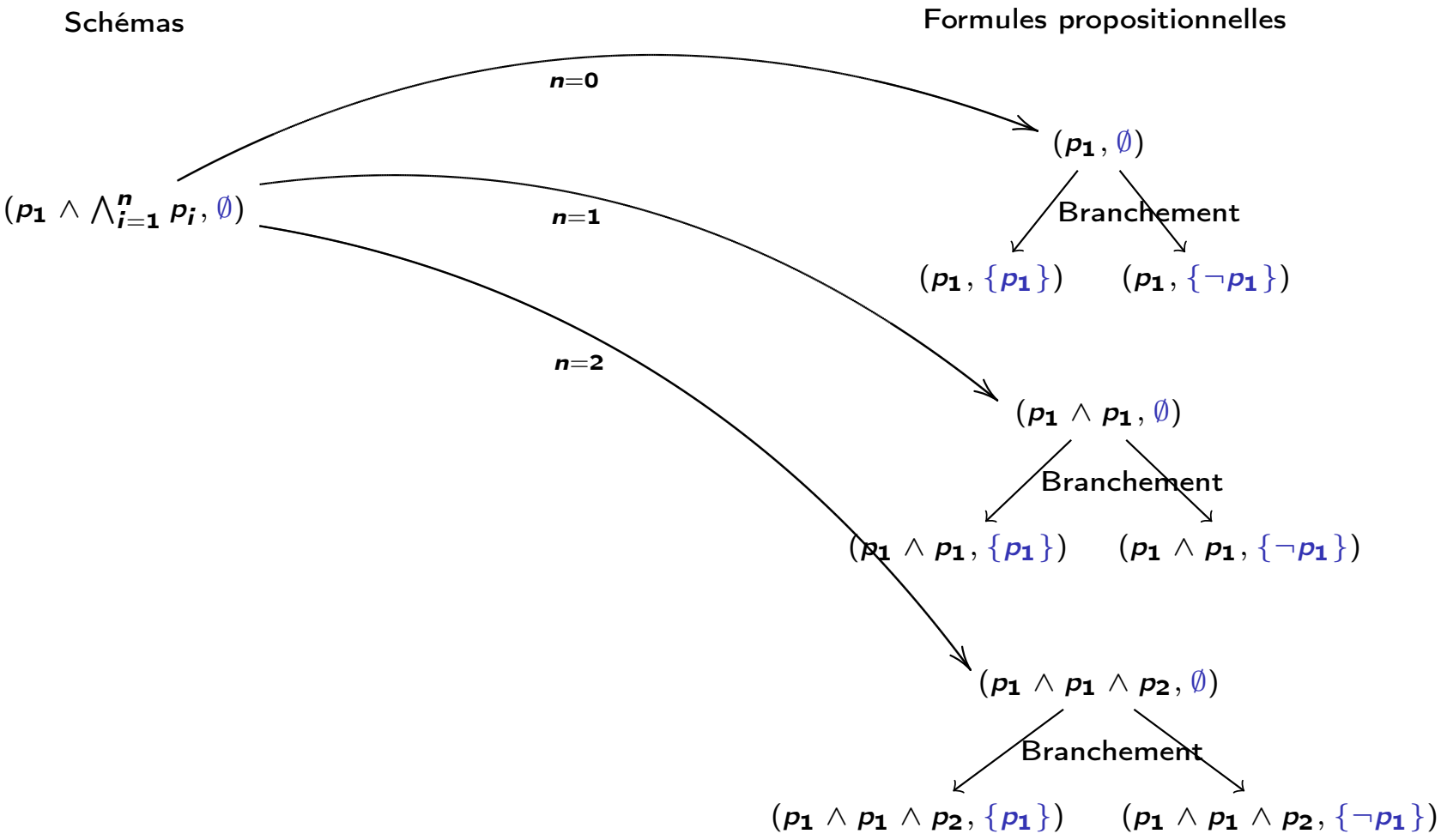
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (1/3)



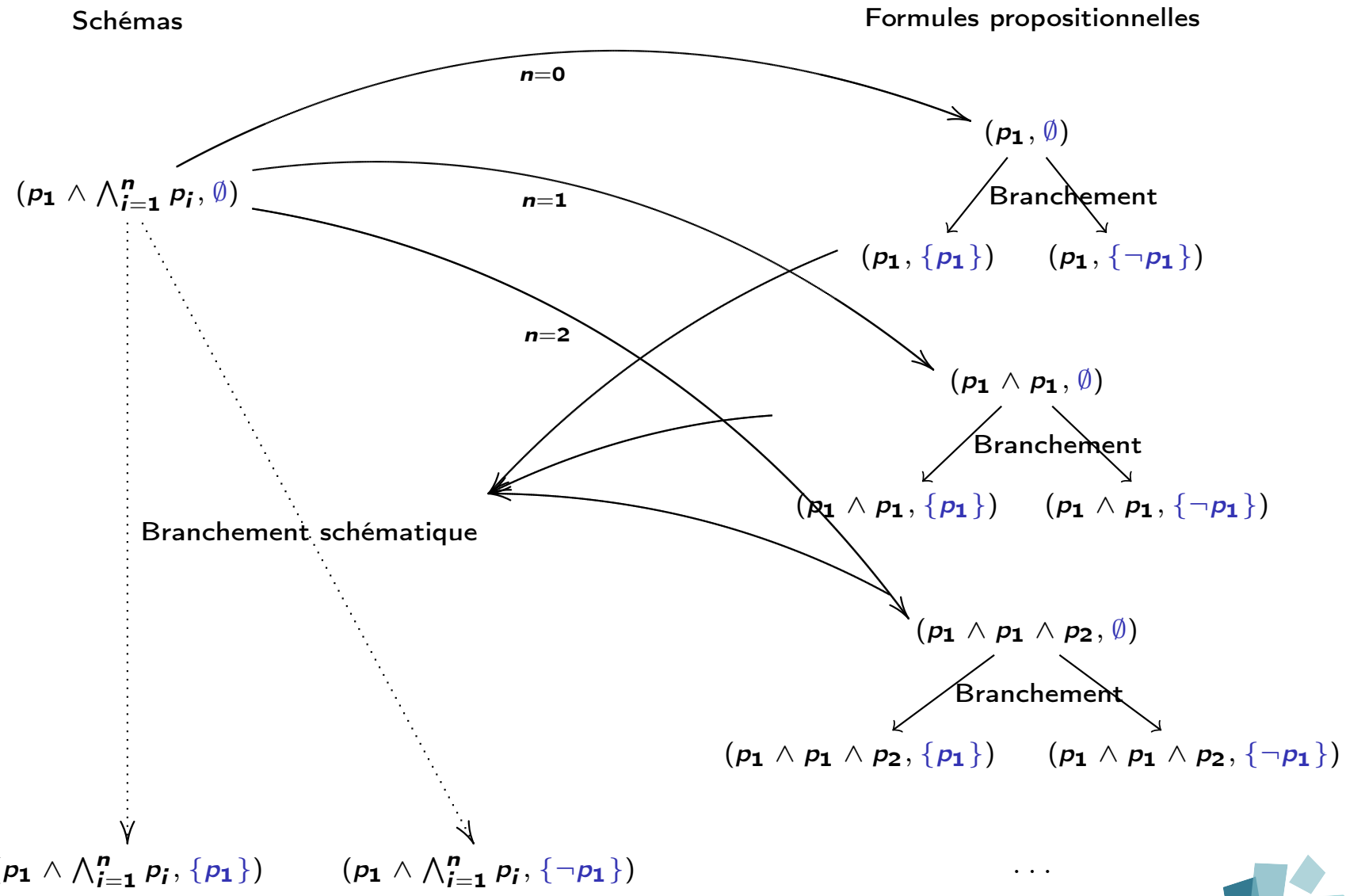
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (1/3)



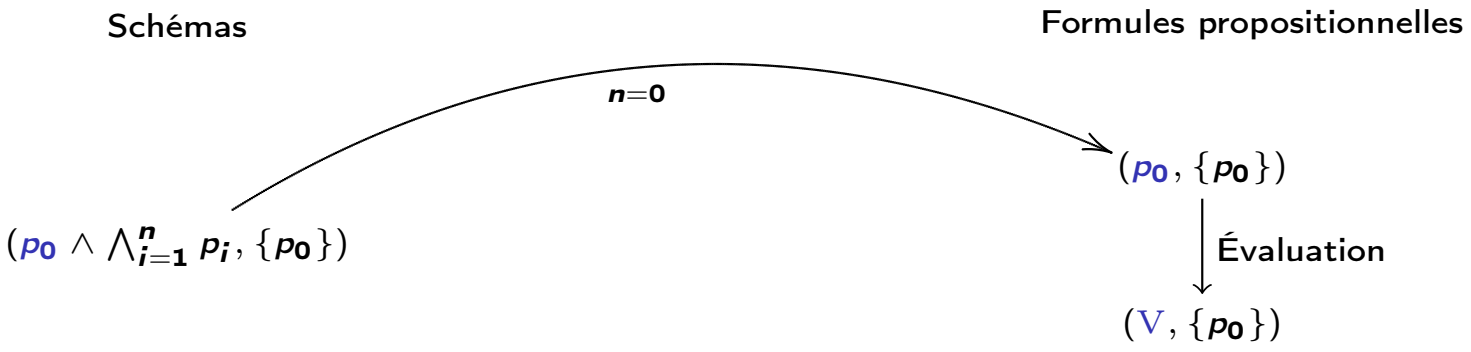
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (1/3)



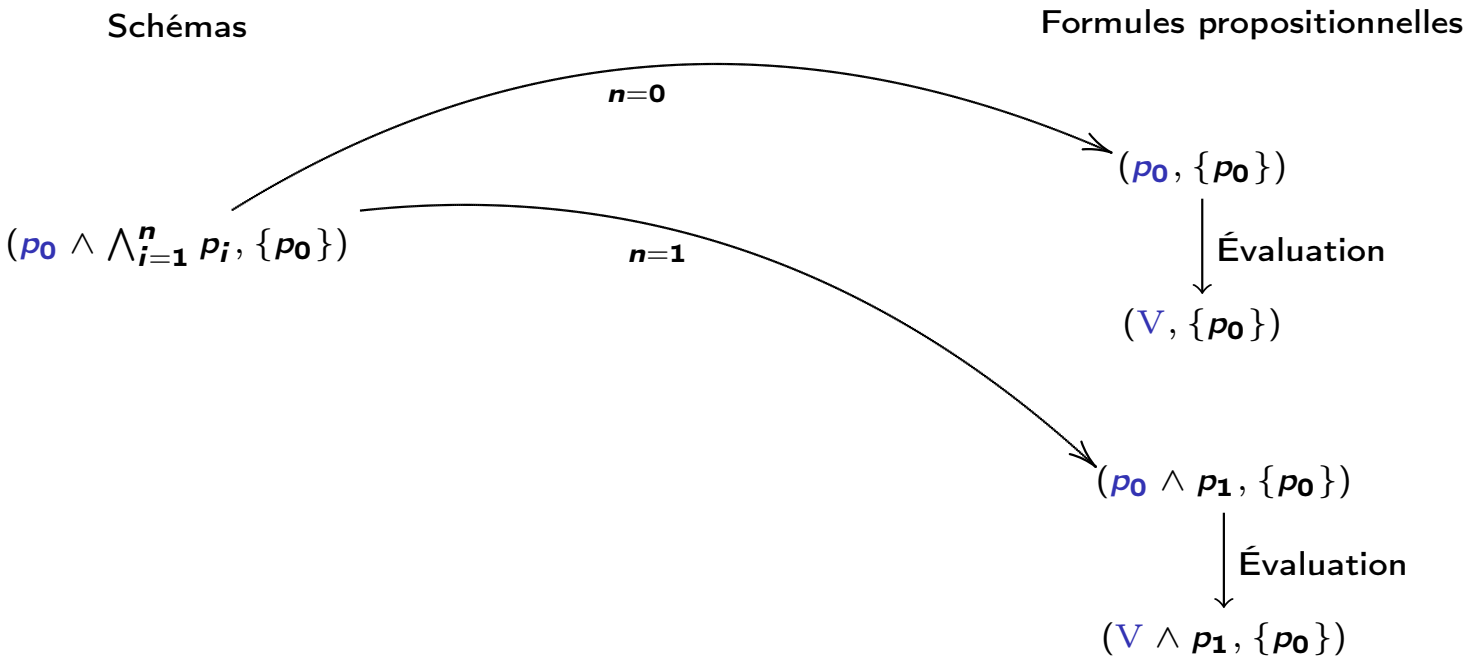
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (1/3)



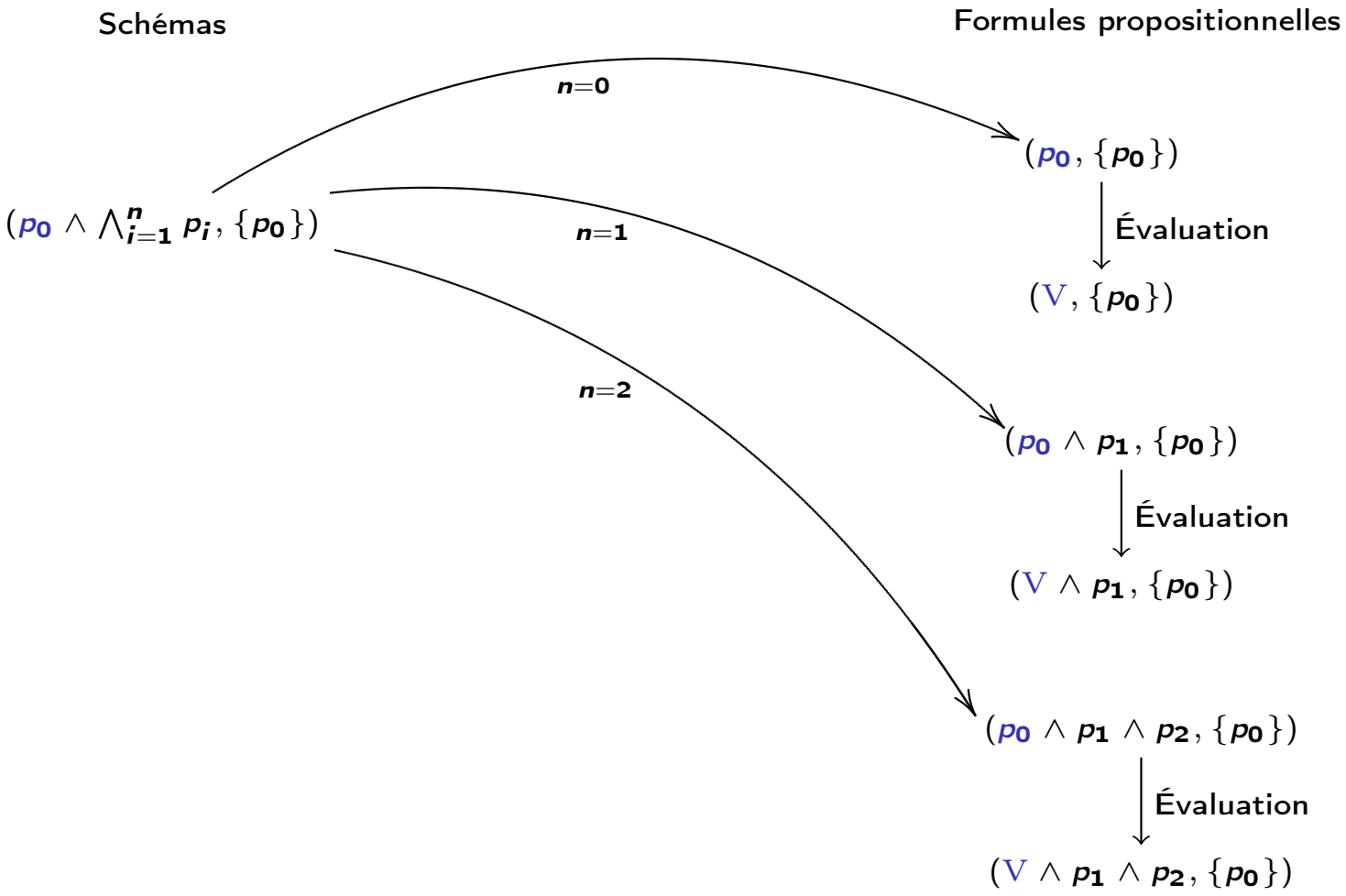
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (2/3)



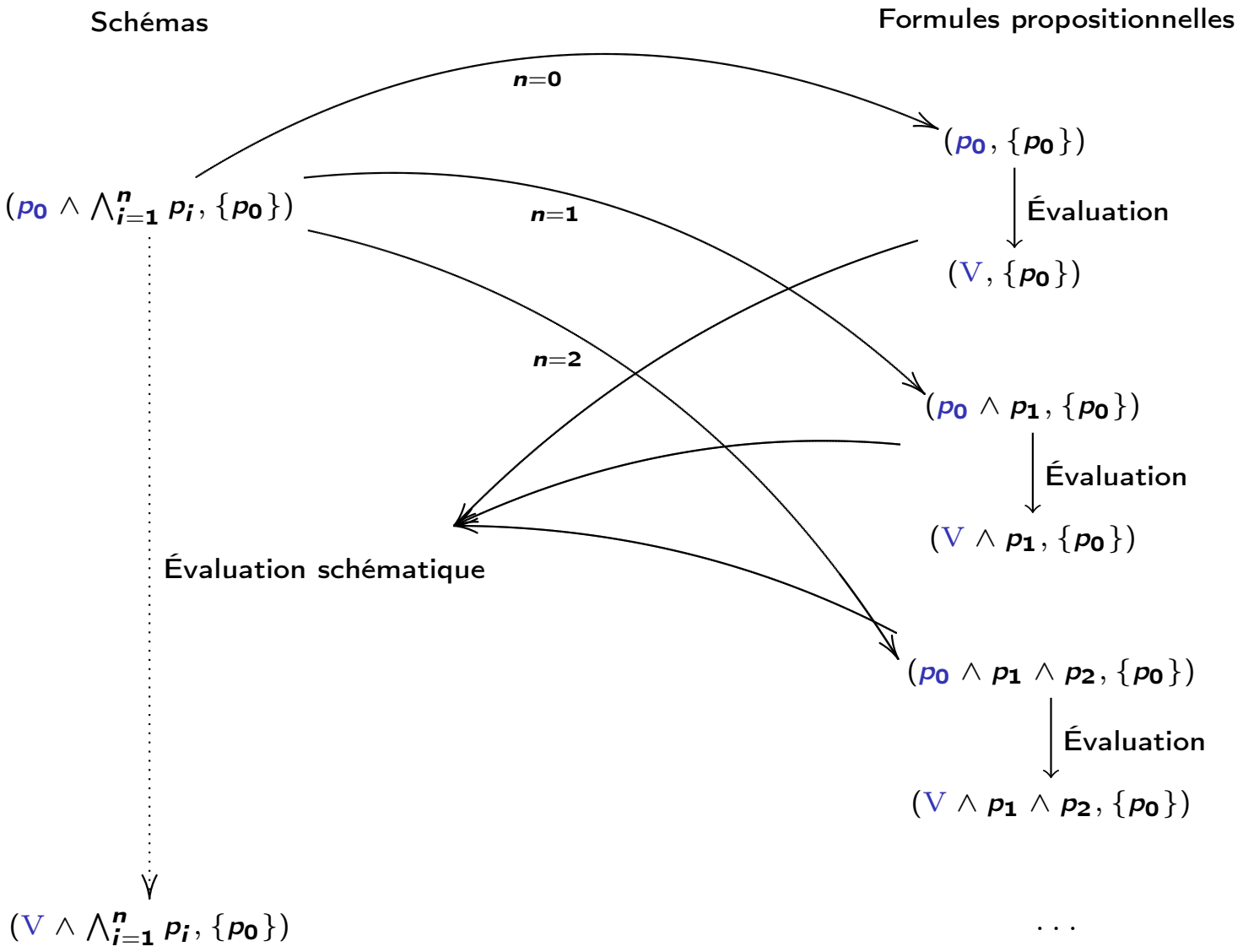
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (2/3)



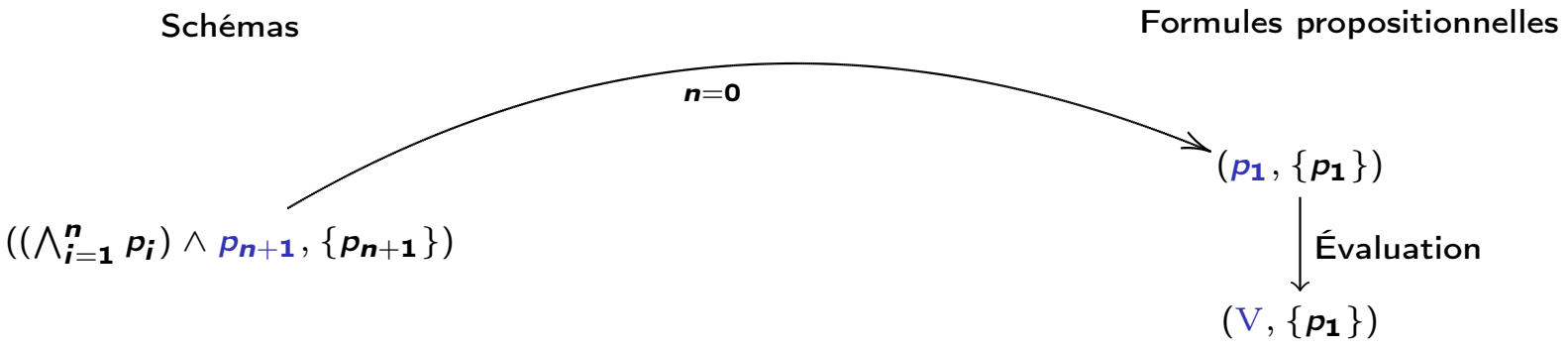
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (2/3)



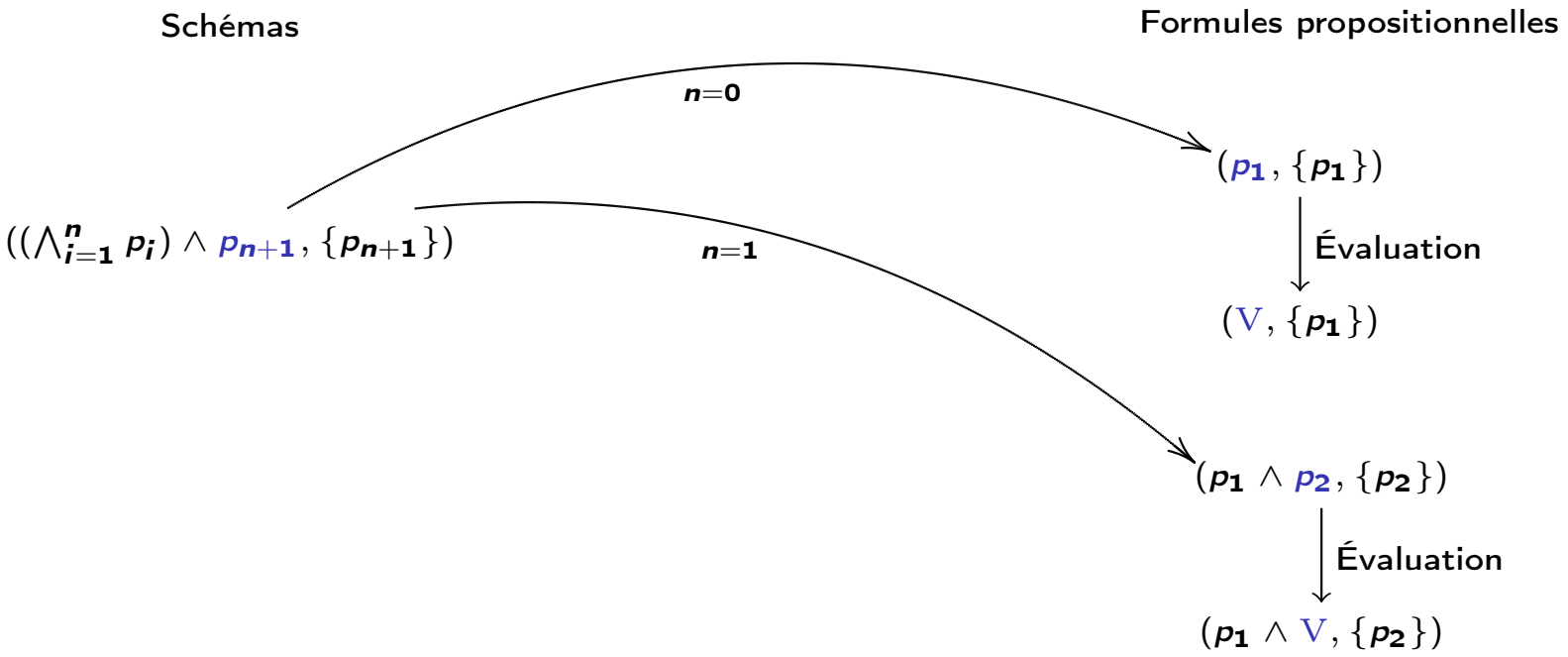
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (2/3)



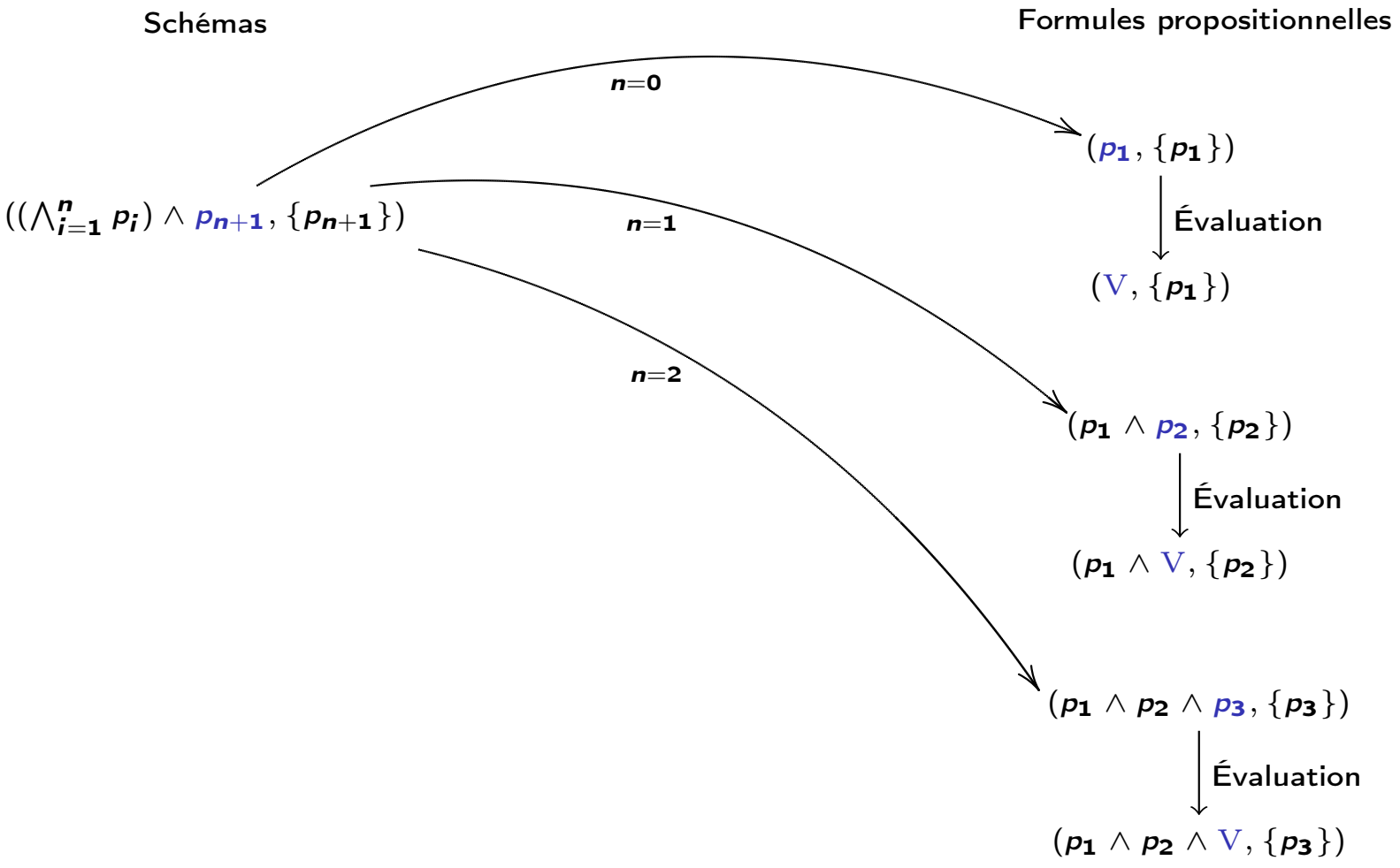
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (3/3)



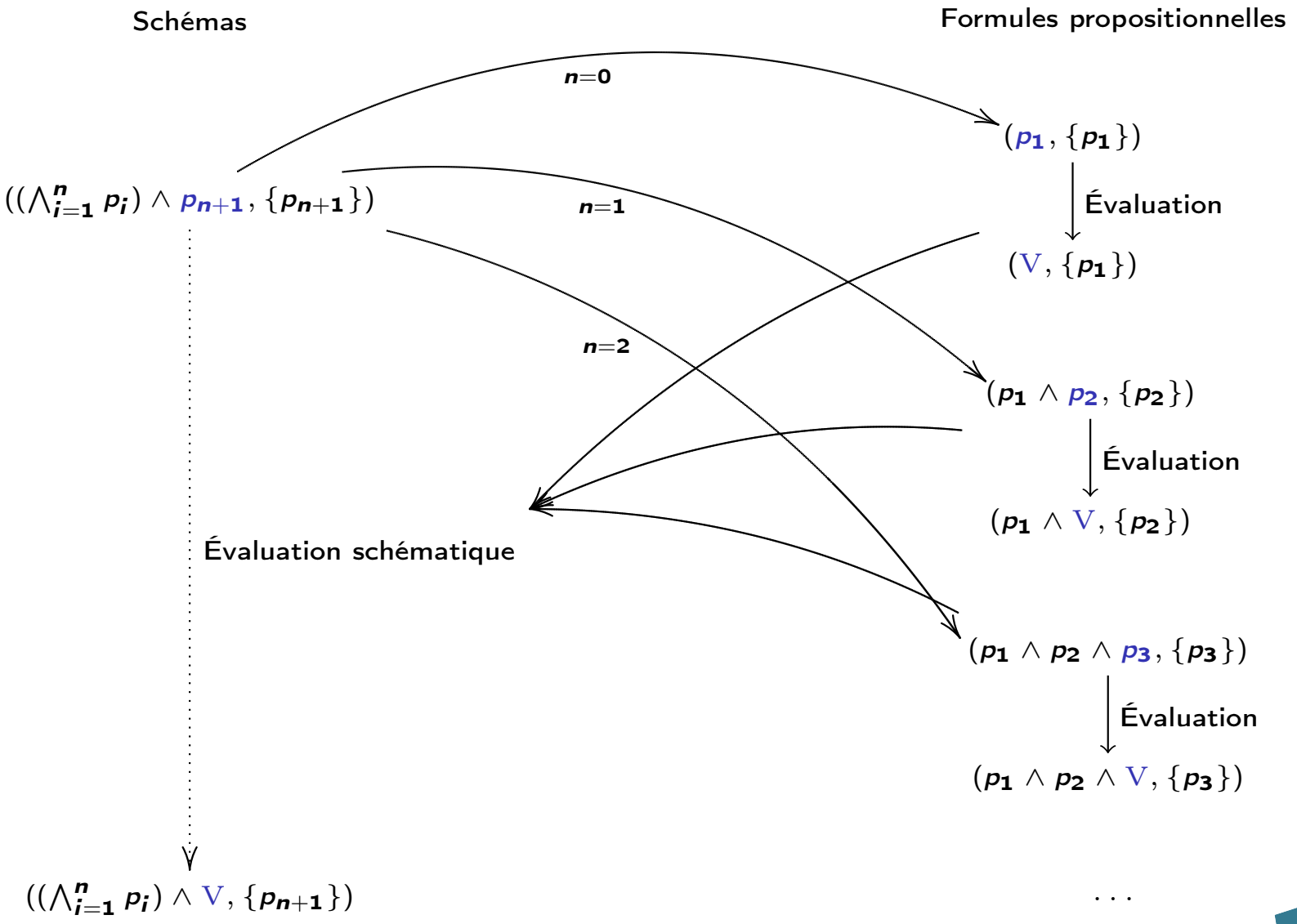
Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (3/3)



Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (3/3)



Extension aux schémas de formules : application parallèle, factorisée et paresseuse de DPLL sur toutes les instances (3/3)



Extension aux schémas de formules : formalisation

Première généralisation = les règles restent les mêmes :

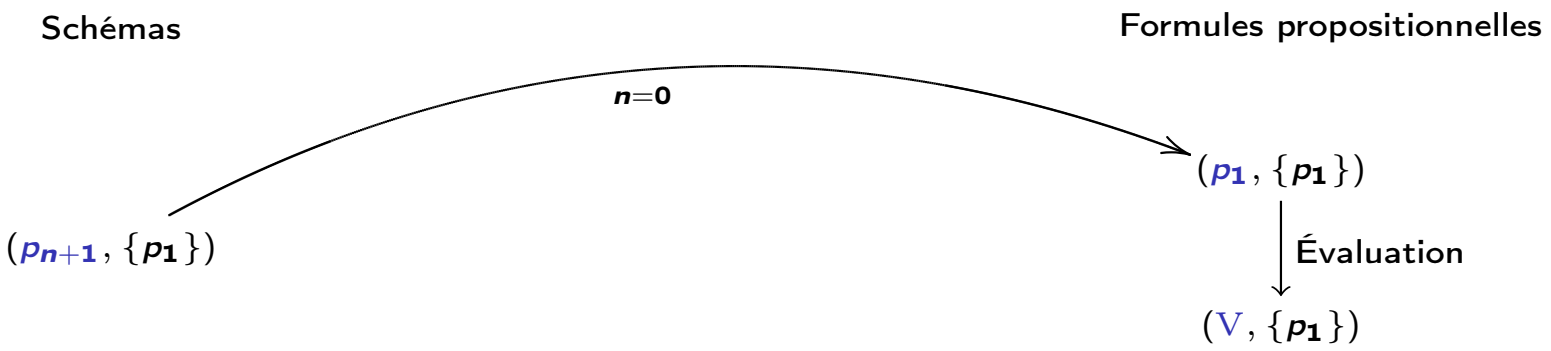
Branchement	Évaluation (pos.)	Évaluation (neg.)
(ϕ, L) $\swarrow \quad \searrow$ $(\phi, L \cup \{p_e\}) \qquad (\phi, L \cup \{\neg p_e\})$ pour une variable prop. p_e de ϕ	$(\phi[p_e], L \cup \{p_e\})$ $\downarrow$ $(\phi[\mathrm{V}], L \cup \{p_e\})$	$(\phi[p_e], L \cup \{\neg p_e\})$ $\downarrow$ $(\phi[\mathrm{F}], L \cup \{\neg p_e\})$

À la fin de la construction :

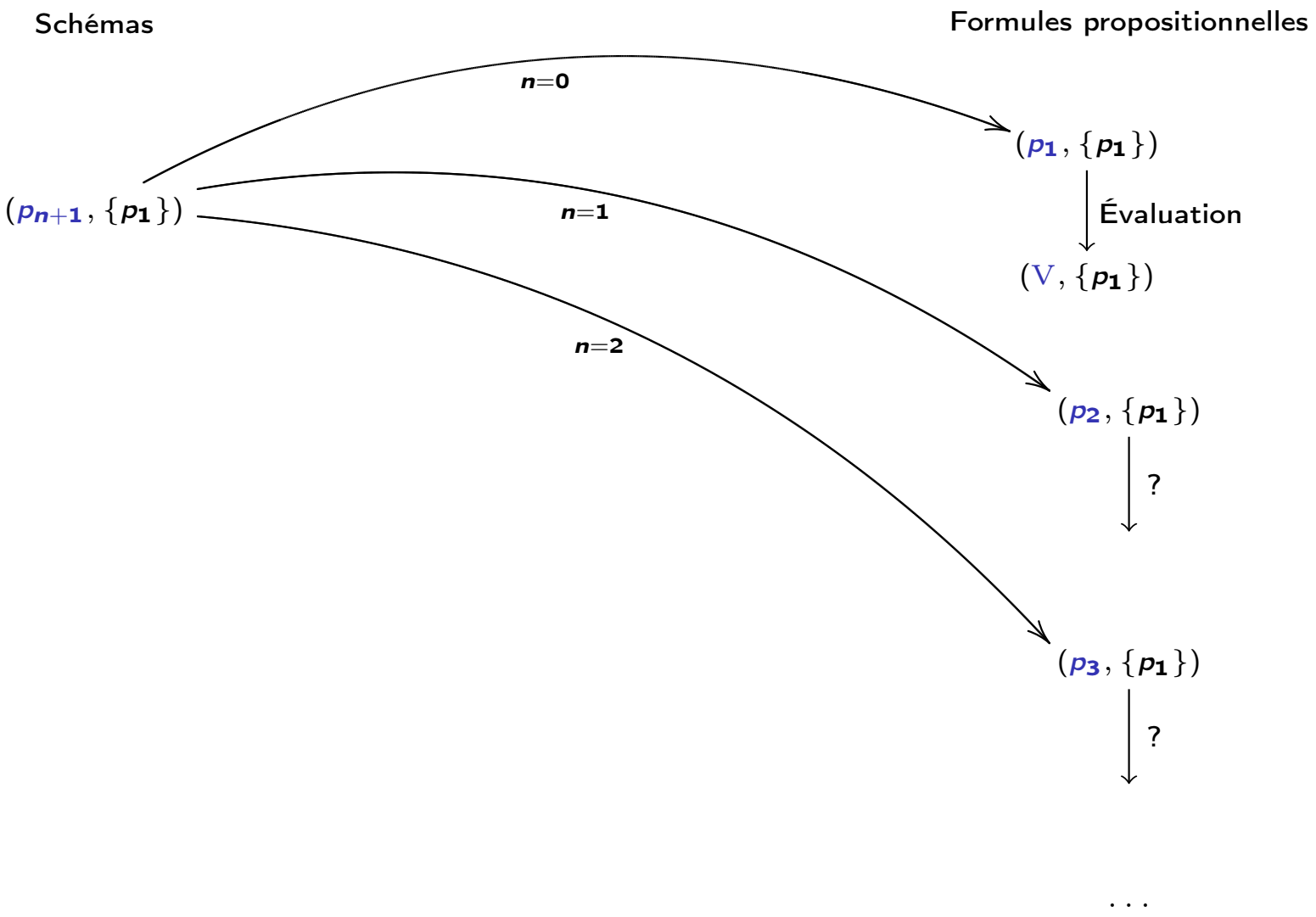
- une feuille étiquetée par $\mathrm{V} \Rightarrow$ SAT
- toutes les feuilles étiquetées par $\mathrm{F} \Rightarrow$ INSAT

Mais les feuilles peuvent être étiquetées par autre chose que F *ou* V *(p.ex. aucune règle ne s'applique à* $\bigwedge_{i=1}^{n} p_i$*).*

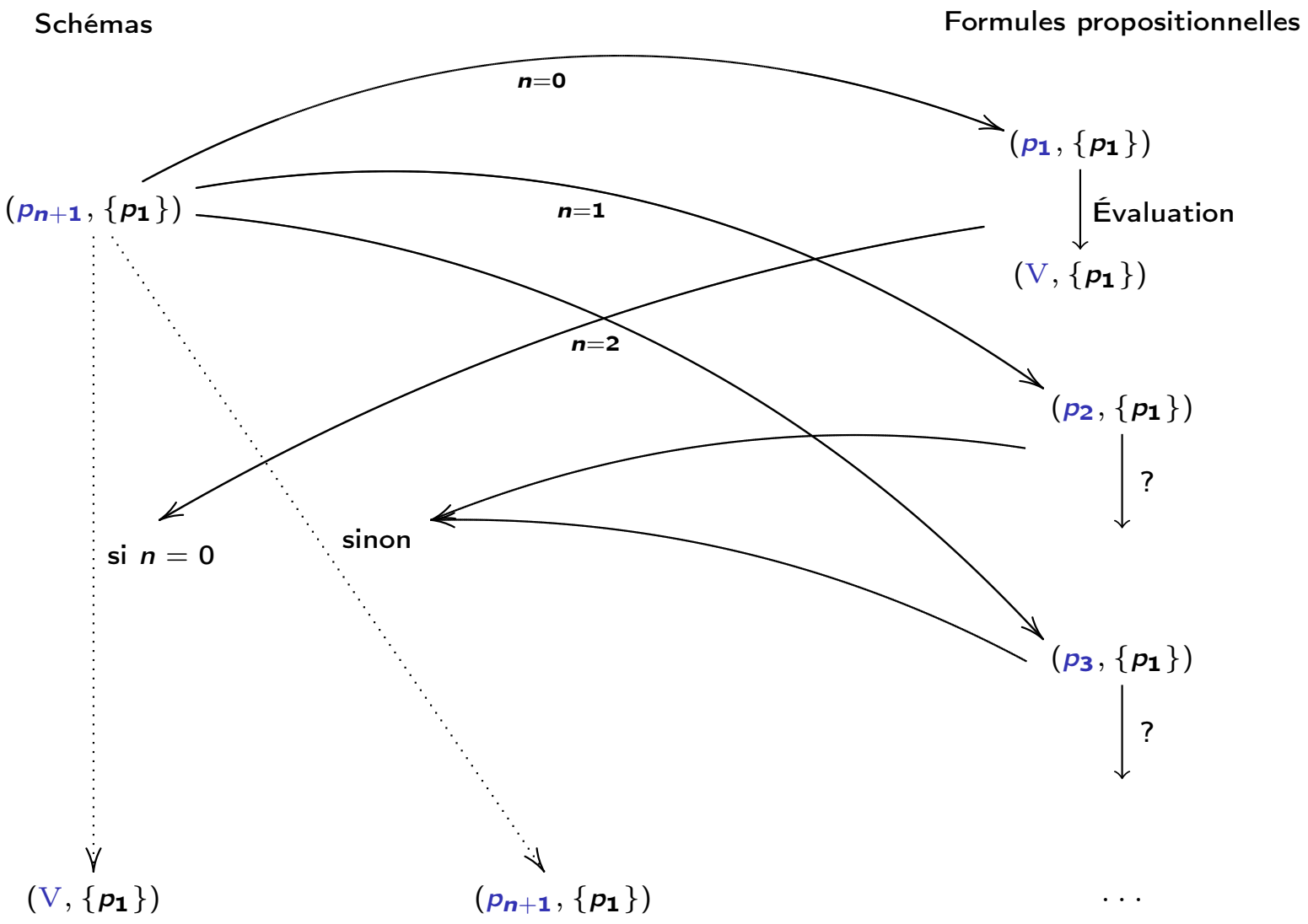
Règles non uniformes : exemple introductif



Règles non uniformes : exemple introductif



Règles non uniformes : exemple introductif



Règles non uniformes : formalisation

Seconde généralisation = nœuds aussi étiquetés par une contrainte :

Branchement :

$$(\phi, c, L) \to (\phi, c, L \cup \{\neg p_e\}) \qquad (\phi, c, L \cup \{p_k\})$$

Évaluation (pos.) :

$$(\phi[p_e], c, L \cup \{p_f\}) \to (\phi[\mathrm{V}], c \wedge e = f, L \cup \{p_f\}) \qquad (\phi[p_e], c \wedge e \neq f, L \cup \{p_f\})$$

Évaluation (neg.) :

$$(\phi[p_e], c, L \cup \{\neg p_f\}) \to (\phi[\mathrm{F}], c \wedge e = f, L \cup \{\neg p_f\}) \qquad (\phi[p_e], c \wedge e \neq f, L \cup \{\neg p_f\})$$

Quand une feuille est étiquetée par une contrainte insatisfaisable, on arrête de la développer.

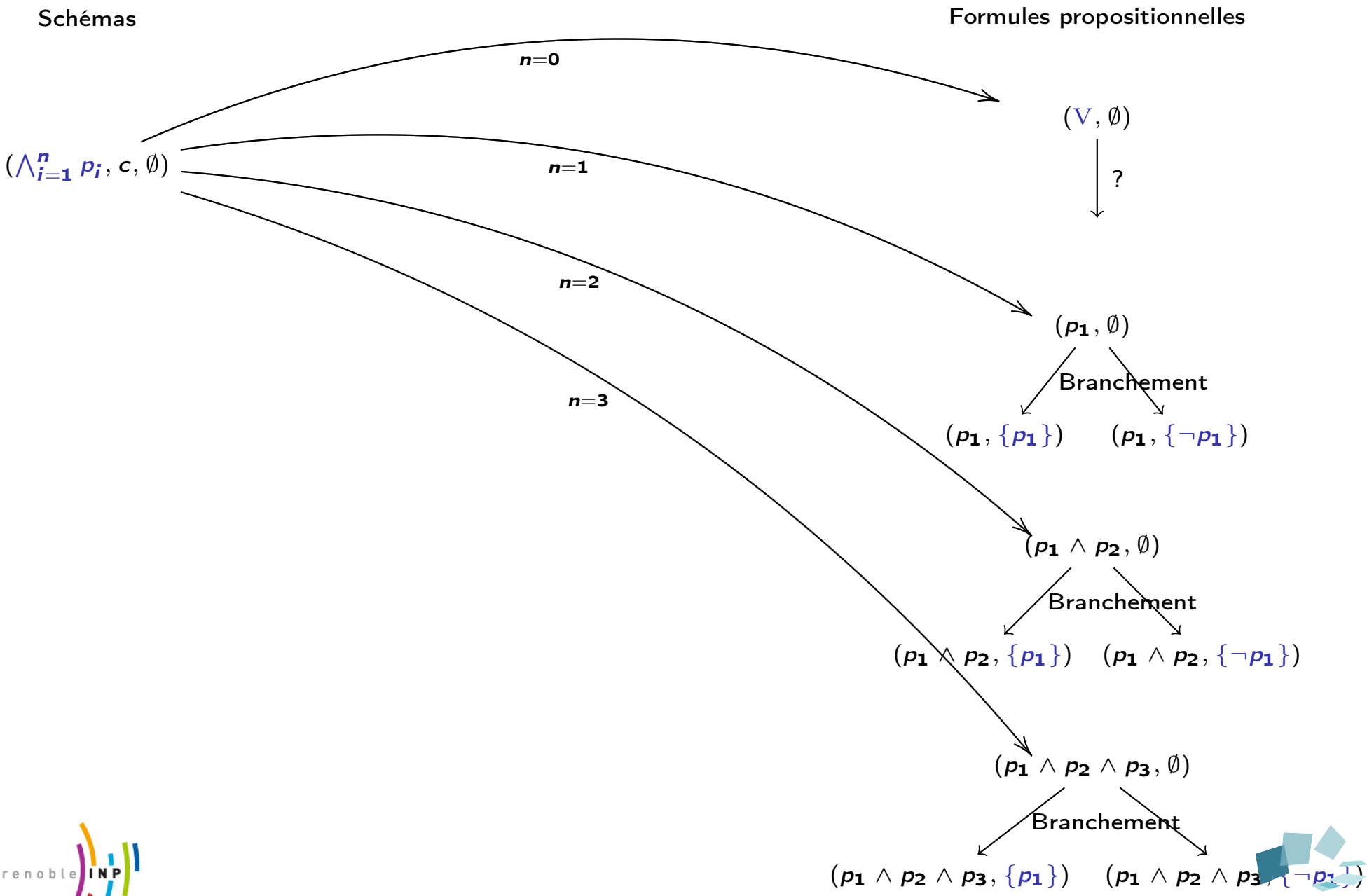
À la fin de la construction :

- une feuille étiquetée par V $\Rightarrow$ SAT
- toutes les feuilles étiquetées par F *ou une contrainte insatisfaisable* $\Rightarrow$ INSAT

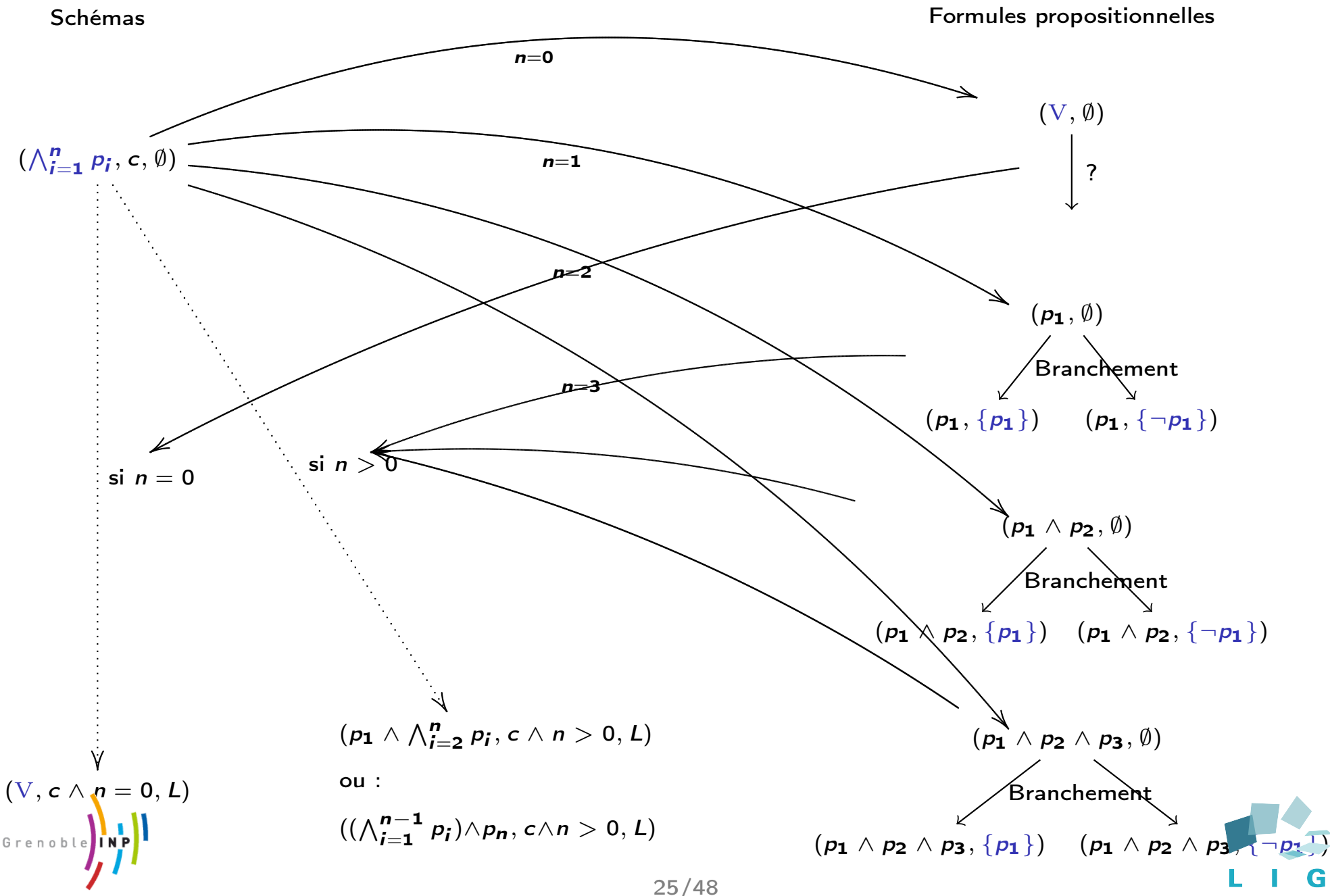
Traitement des itérations : motivation

$$\begin{tikzcd}
\textbf{Schémas} & & \textbf{Formules propositionnelles} \\
(\bigwedge_{i=1}^{n} p_i, c, \emptyset) \arrow[rr, bend left, "n=0"] & & (\mathrm{V}, \emptyset) \arrow[d, "?"] \\
& & {}
\end{tikzcd}$$

Traitement des itérations : motivation



Traitement des itérations : motivation



Traitement des itérations : formalisation

Troisième généralisation = ajout d'une règle de « dépliage » :

Branchement :

$$(\phi, c, L) \rightarrow (\phi, c, L \cup \{\neg p_e\}) \quad (\phi, c, L \cup \{p_k\})$$

Évaluation (pos.) :

$$(\phi[p_e], c, L \cup \{p_f\}) \rightarrow (\phi[\mathrm{V}], c \wedge e = f, L \cup \{p_f\}) \quad (\phi[p_e], c \wedge e \neq f, L \cup \{p_f\})$$

Évaluation (neg.) :

$$(\phi[p_e], c, L \cup \{\neg p_f\}) \rightarrow (\phi[\mathrm{F}], c \wedge e = f, L \cup \{\neg p_f\}) \quad (\phi[p_e], c \wedge e \neq f, L \cup \{\neg p_f\})$$

Dépliage :

$$(\bigwedge_{i=1}^{n} \phi', c, L) \rightarrow (\mathrm{V}, c \wedge n = 0, L) \quad ((\bigwedge_{i=1}^{n-1} \phi') \wedge \phi'[n/i], c \wedge n > 0, L)$$

+ dual pour $\bigvee_{i=1}^{n} \phi'$.

$\rightarrow$ « DPLL* » simplifié (pas d'évaluation sous les itérations)

Propriétés de DPLL*

- quand la construction de l'arbre termine chaque feuille est dans l'une des situations suivantes :
  1. elle est étiquetée par V
  2. elle est étiquetée par F
  3. elle est étiquetée par une contrainte insatisfaisable
- une feuille étiquetée par V $\Rightarrow$ schéma initial satisfaisable
- sinon, (c.-à-d. toutes les feuilles sont dans 2 ou 3) $\Rightarrow$ schéma initial insatisfaisable
- la construction de l'arbre ne termine pas en général

Exemple de non terminaison

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$$

$\neg p_{n+1}$

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \mathrm{V}$$

$$\|$$

$$p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1})$$

$n = 0$

$$p_1 \wedge \mathrm{V}$$

$$\|$$

$$p_1$$

$n + 1 = 1$

$$\mathrm{F}$$

$n + 1 \neq 1$

contradictoire

$n > 0$

$$p_1 \wedge \bigwedge_{i=1}^{n-1} (p_i \Rightarrow p_{i+1}) \wedge (p_n \Rightarrow p_{n+1})$$

$$\downarrow$$

$$p_1 \wedge \bigwedge_{i=1}^{n-1} (p_i \Rightarrow p_{i+1}) \wedge (p_n \Rightarrow \mathrm{F})$$

$$\|$$

$$p_1 \wedge \bigwedge_{i=1}^{n-1} (p_i \Rightarrow p_{i+1}) \wedge \neg p_n$$

$$\downarrow$$

$$\ldots$$

$$(\infty)$$

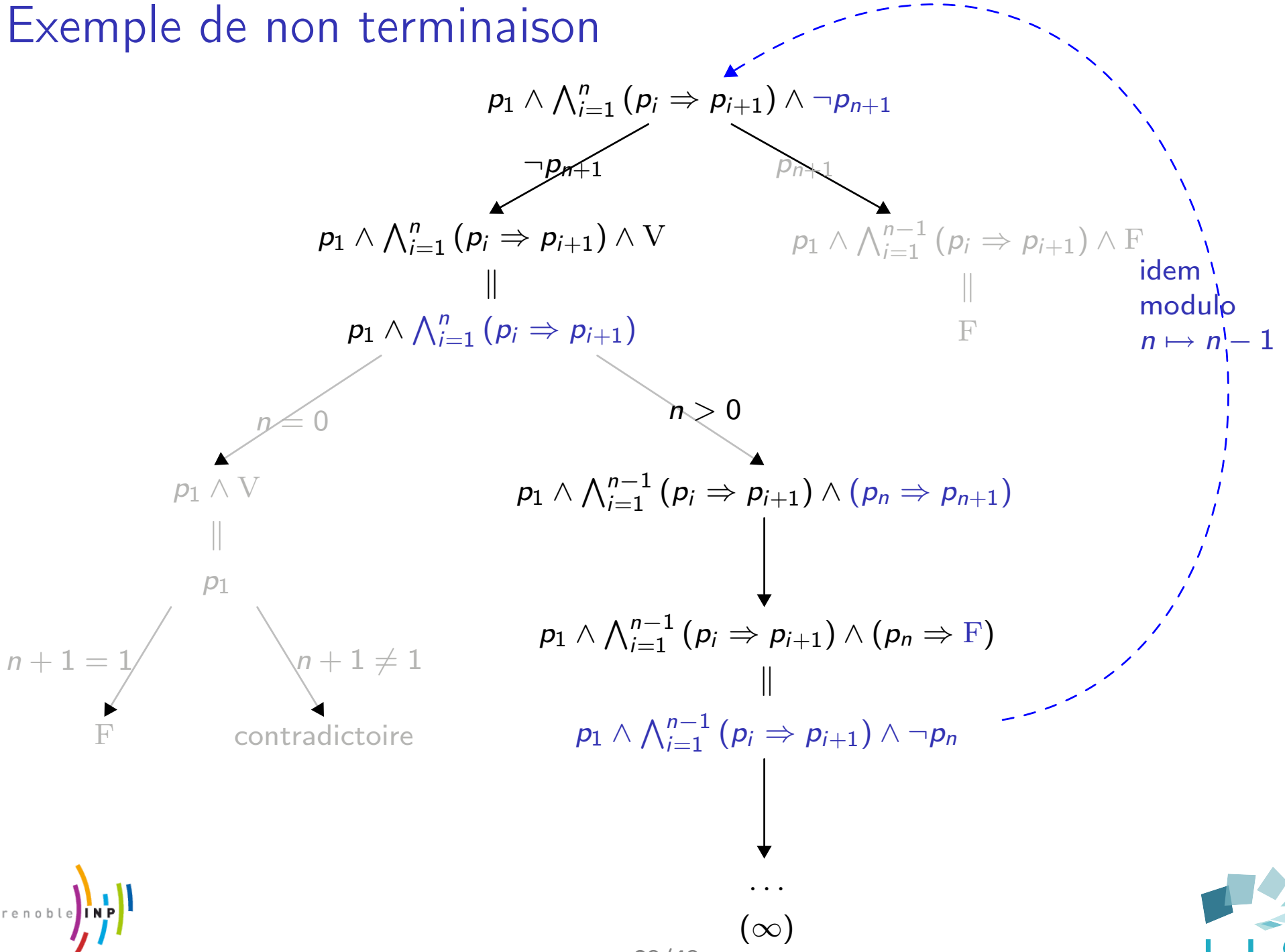
p_{n+1}

$$p_1 \wedge \bigwedge_{i=1}^{n-1} (p_i \Rightarrow p_{i+1}) \wedge \mathrm{F}$$

$$\|$$

$$\mathrm{F}$$

Exemple de non terminaison



Comment empêcher la non terminaison ?

- ex. précédent : on a conclu qu'il n'y avait pas terminaison en observant un *cycle*
- idée : intégrer à l'algorithme une *détection de cycle*
- nécessité d'une définition formelle de « cycle »
- ex. précédent :

  Définition :
  il y a un cycle lorsque l'étiquette d'une feuille est égale à celle d'un nœud déjà vu, modulo la substitution $n \mapsto n - 1$.

- plus généralement, on considère n'importe quelle substitution $n \mapsto n - k$ pour un quelconque $k \in \mathbb{N}$, $k \geq 1$.

À propos de la définition de cycle

- autres définitions possibles
- Exemple :
  Définition précédente $\Rightarrow$ *cycle entre* $\bigwedge_{i=1}^{n-1} p_i$ *et* $\bigwedge_{i=1}^{n} p_i$
  mais pas entre $\bigwedge_{i=2}^{n} p_i$ *et* $\bigwedge_{i=1}^{n} p_i$
- définitions plus générales possibles, p.ex. déf. sémantique :

  Définition :
  Il y a un cycle entre une feuille et un nœud déjà vu si pour tout modèle $\mathfrak{M}_1$ *de la feuille il existe un modèle* $\mathfrak{M}_2$ *du nœud tel que* $\mathfrak{M}_2 \prec \mathfrak{M}_1$ *pour un ordre bien fondé* $\prec$ *sur les interprétations.*
- $\Rightarrow$ mais détection de cycle indécidable.
  Aucune déf. ne peut assurer la terminaison, *(indécidabilité)*.
- on garde la définition précédente facile à implémenter, plus restrictive et suffisante pour nos résultats

Ajout de la détection de cycle

Branchement :

$$(\phi, c, L) \rightarrow (\phi, c, L \cup \{\neg p_e\}) \quad (\phi, c, L \cup \{p_k\})$$

Évaluation (pos.) :

$$(\phi[p_e], c, L \cup \{p_f\}) \rightarrow (\phi[\mathrm{V}], c \wedge e = f, L \cup \{p_f\}) \quad (\phi[p_e], c \wedge e \neq f, L \cup \{p_f\})$$

Évaluation (neg.) :

$$(\phi[p_e], c, L \cup \{\neg p_f\}) \rightarrow (\phi[\mathrm{F}], c \wedge e = f, L \cup \{\neg p_f\}) \quad (\phi[p_e], c \wedge e \neq f, L \cup \{\neg p_f\})$$

Dépliage :

$$(\bigwedge_{i=1}^{n} \phi', c, L) \rightarrow (\mathrm{V}, c \wedge n = 0, L) \quad ((\bigwedge_{i=1}^{n-1} \phi') \wedge \phi'[n/i], c \wedge n > 0, L)$$

$\boxed{+ \text{ détection de cycle :}}$

Si un cycle est détecté, on arrête de développer la branche.

Plan

Introduction générale : la déduction automatique

Introduction à la thèse : les schémas de formules

Déduction automatique et schémas de formules

Analyse d'un point technique : décidabilité des schémas réguliers

Conclusion

Un ensemble de schémas pour lesquels DPLL* termine : les schémas réguliers

$$\begin{array}{lcl} \phi & ::= & \phi_1 \vee \phi_2 \,|\, \phi_1 \wedge \phi_2 \,|\, \bigwedge_{i=1}^{n-k} \phi' \,|\, \bigvee_{i=1}^{n-k} \phi' \,|\, p_{\underline{n+k}} \,|\, \neg p_{\underline{n+k}} \\ \phi' & ::= & \phi'_1 \vee \phi'_2 \,|\, \phi'_1 \wedge \phi'_2 \,|\, p_{\underline{i+k}} \,|\, \neg p_{\underline{i+k}} \end{array}$$

où $k \in \mathbb{Z}$ et i, n variables entières

Propriétés importantes :

- itérations non imbriquées
- indices contiennent au plus une variable

Exemple :

- $p_1 \wedge \bigwedge_{i=1}^{n} (p_i \Rightarrow p_{i+1}) \wedge \neg p_{n+1}$ *régulier*
- $\bigwedge_{i=1}^{n} p_i \wedge \bigvee_{i=1}^{n} \neg p_i$ *régulier*
- $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} (p_i \vee q_j)$ *pas régulier*
- $\bigwedge_{i=1}^{n} p_{i+n}$ *pas régulier*
- $\bigwedge_{i=1}^{n} p_{i+i}$ *pas régulier*

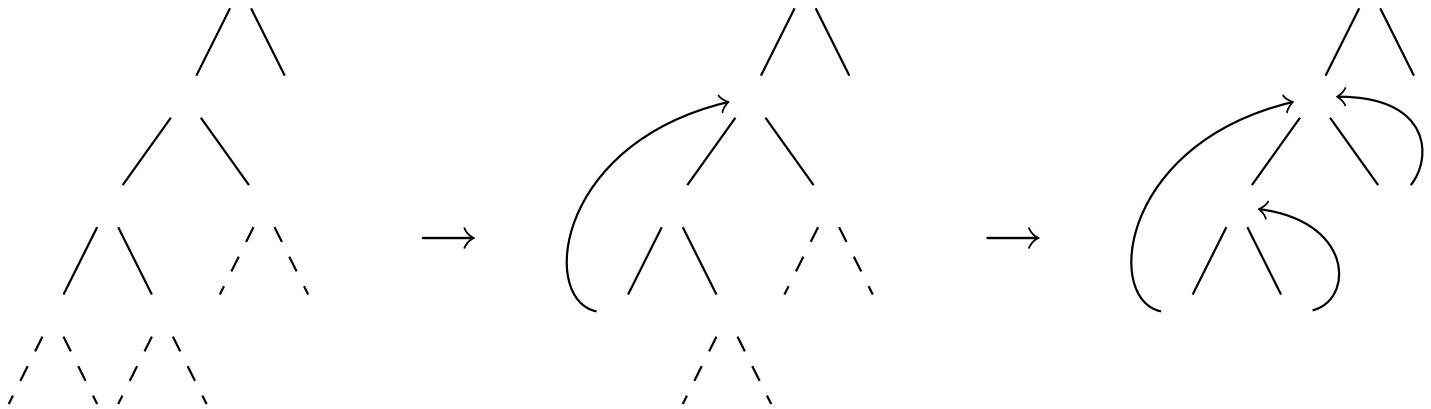
Que peut-on exprimer avec les schémas réguliers ?

- circuits :
  - additionneur à propagation de retenue
  - additionneur à retenue anticipée
  - comparateur
  - en fait, toute l'arithmétique linéaire sur les vecteurs de bits
  - ...

  $\longrightarrow$ circuits dont la structure est une répétition séquentielle d'un circuit élémentaire.
- automates :
  - appartenance
  - vide
  - inclusion
  - ...

Comment démontrer que DPLL* termine : principe général

On veut assurer qu'il y aura toujours un cycle :



$\longrightarrow$ Montrons que :

ensemble des formules générées par DPLL* = fini à la relation de « cycle » près (c.-à-d. à une substitution $n \mapsto n - k$ près pour un $k \in \mathbb{N}$, $k \geq 1$)

(Esquisse de) preuve de terminaison : stratégie

$$
\begin{tikzcd}
(\bigwedge_{i=1}^{n-k} \phi, c, L) \arrow[d, "\text{Dépliage}"'] \\
((\bigwedge_{i=1}^{n-k-1} \phi) \wedge \psi, c', L) \arrow[d, "\text{Branchement}^*"'] \\
((\bigwedge_{i=1}^{n-k-1} \phi) \wedge \psi, c', L \cup \underbrace{\{l_1, l_2, \ldots, l_k\}}_{\text{littéraux de } \psi}) \arrow[d, "\text{Évaluation}^*"'] \\
(\bigwedge_{i=1}^{n-k-1} \phi, c'', L \cup \{l_1, l_2, \ldots, l_k\}) \arrow[uu, bend left]
\end{tikzcd}
$$

Preuve de terminaison : divergence

Il y a clairement divergence :

$$
\begin{array}{c}
(\bigwedge_{i=1}^{n-k} \phi, c, L) \\
\dots \downarrow \\
(\bigwedge_{i=1}^{n-k-1} \phi, c', L \cup \{l_1^1, l_2^1, \dots, l_k^1\}) \\
\dots \downarrow \\
(\bigwedge_{i=1}^{n-k-2} \phi, c'', L \cup \{l_1^1, l_2^1, \dots, l_k^1, l_1^2, l_2^2, \dots, l_k^2\}) \\
\dots \downarrow \\
(\bigwedge_{i=1}^{n-k-3} \phi, c''', L \cup \{l_1^1, l_2^1, \dots, l_k^1, l_1^2, l_2^2, \dots, l_k^2, l_1^2, l_2^3, \dots, l_k^3\}) \\
\dots \downarrow \\
\dots
\end{array}
$$

(pour simplifier, on ne tient pas compte des contraintes)

Mais beaucoup de littéraux inutiles car « purs »...

Preuve de terminaison : notion de littéral pur

Définition : (cas propositionnel)

Un littéral est pur dans une formule propositionnelle ssi son complémentaire n'apparaît pas dans la formule.

Définition : (cas schématique)

Un littéral est pur dans un schéma ssi il est propositionnellement pur pour toute valeur de n.

Exemple :

p_{n+2} *est pur dans* $\bigwedge_{i=1}^{n} p_i \Rightarrow p_{i+1}$.

Théorème :

La satisfaisabilité est préservée quand on évalue un littéral pur à V.

Exemple :

Nœud étiqueté par $(\bigwedge_{i=1}^{n} p_i \Rightarrow p_{i+1}, c, \{p_{n+2}\})$
$\Rightarrow$ *idem de considérer* $(\bigwedge_{i=1}^{n} p_i \Rightarrow p_{i+1}, c, \emptyset)$.

Ajout de la règle qui évalue un littéral pur à V.

Preuve de terminaison : élimination des littéraux purs

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \wedge \phi[n-1/i] \; l_1^1 \ldots l_k^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \quad l_1^2 \ldots l_k^2 \quad l_1^1 \ldots l_k^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \quad l_1^3 \ldots l_k^3 \quad l_1^2 \ldots l_k^2 \quad l_1^1 \ldots l_k^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \quad l_1^4 \ldots l_k^4 \quad l_1^3 \ldots l_k^3 \quad l_1^2 \ldots l_k^2 \quad l_1^1 \ldots l_k^1$$

$\downarrow$(élim. des lits purs)

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \wedge \phi[n-1/i] \; l_1^1 \ldots l_k^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \quad l_1^2 \ldots l_k^2$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \quad l_1^3 \ldots l_k^3$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \quad l_1^4 \ldots l_k^4$$

$\downarrow$(subst. de la forme $n \mapsto n-k$)

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{1\prime} \ldots l_k^{1\prime}$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{2\prime} \ldots l_k^{2\prime}$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{3\prime} \ldots l_k^{3\prime}$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{4\prime} \ldots l_k^{4\prime}$$

k borné $\Rightarrow$ quantité finie de possibilités

Preuve de terminaison : preuve que k est borné

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \quad l_1^1 \ldots l_k^1$$

$\downarrow$explicitation et tri des littéraux

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \quad \neg p_{n+1} p_{n+2} p_{n+3} \neg p_{n+4}$$

- montrer qu'il existe $k_{\max} \in \mathbb{N}$ t.q. p_{n+k} est pur pour tout $k \geq k_{\max}$:

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \quad \neg p_{n+1} \ldots \neg p_{n+k_{\max}-1} \cancel{p_{n+k_{\max}}} \cancel{p_{n+k_{\max}+1}} \cancel{\neg p_{n+k_{\max}+2}} \ldots$$

  $\Rightarrow$ indices des littéraux non purs $\in$ intervalle borné par rapport à n

- ϕ contient un atome p_{i+k} ($k \in \mathbb{Z}$ par déf. des schémas réguliers) :

$$(\ldots p_{i+k}[1/i] \ldots) \wedge \cdots \wedge (\ldots p_{i+k}[n-1/i] \ldots) \wedge (\ldots p_{i+k}[n/i] \ldots) \quad \neg p_{n+1} p_{n+2} p_{n+3} \neg p_{n+4}$$

  supposons k maximal parmi indices de la forme $i+k$

- p_{n+k-1} ou p_{n+k} peuvent apparaître dans l'itération :

$$(\ldots p_{i+k}[1/i] \ldots) \wedge \cdots \wedge (\ldots \; p_{i+k}[n-1/i] \; \ldots) \wedge (\ldots \; p_{i+k}[n/i] \; \ldots) \qquad \ldots \neg p_{n+k-1} \quad p_{n+k} \quad \ldots$$

- mais pas p_{n+k+1} :

$$(\ldots p_{i+k}[1/i] \ldots) \wedge \cdots \wedge (\ldots \; p_{i+k}[n-1/i] \; \ldots) \wedge (\ldots \; p_{i+k}[n/i] \; \ldots) \qquad \ldots \neg p_{n+k-1} \quad p_{n+k} \quad \neg p_{n+k+1}$$

- $\Rightarrow \neg p_{n+k+1}$ **pur, ainsi que tout littéral d'indice** $> n+k+1$

Preuve de terminaison : élimination des littéraux purs (bis).

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \wedge \phi[n-1/i] \; l_1^1 \ldots l_{k_{\max}}^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \quad l_1^2 \ldots l_{k_{\max}}^2 \qquad l_1^1 \ldots l_k^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \quad l_1^3 \ldots l_{k_{\max}}^3 \qquad l_1^2 \ldots l_k^2 \qquad l_1^1 \ldots l_k^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \quad l_1^4 \ldots l_{k_{\max}}^4 \qquad l_1^3 \ldots l_k^3 \qquad l_1^2 \ldots l_k^2 \qquad l_1^1 \ldots l_k^1$$

$\downarrow$(élim. des lits purs)

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \wedge \phi[n-1/i] \; l_1^1 \ldots l_{k_{\max}}^1$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \wedge \phi[n-2/i] \quad l_1^2 \ldots l_{k_{\max}}^2$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \wedge \phi[n-3/i] \quad l_1^3 \ldots l_{k_{\max}}^3$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n-4/i] \quad l_1^4 \ldots l_{k_{\max}}^4$$

$\downarrow$(subst. de la forme $n \mapsto n-k$)

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{1\prime} \ldots l_{k_{\max}}^{1\prime}$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{2\prime} \ldots l_{k_{\max}}^{2\prime}$$

$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{3\prime} \ldots l_{k_{\max}}^{3\prime}$$

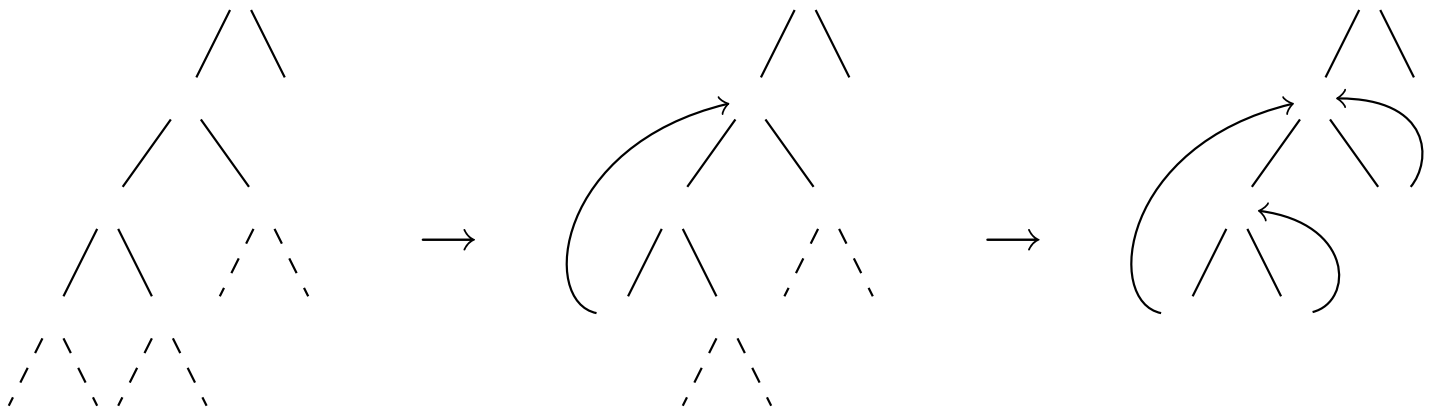
$$\phi[1/i] \wedge \ldots \wedge \phi[n/i] \; l_1^{4\prime} \ldots l_{k_{\max}}^{4\prime}$$

$k_{\max}$ fixe $\Rightarrow$ quantité finie de possibilités

Preuve de terminaison : conclusion.

On a bien montré que l'ensemble des formules générées est fini à une substitution $n \mapsto n - k$ près.

Donc toute branche contient nécessairement un cycle.



Plan

Introduction générale : la déduction automatique

Introduction à la thèse : les schémas de formules

Déduction automatique et schémas de formules

Analyse d'un point technique : décidabilité des schémas réguliers

Conclusion

Bilan des résultats obtenus

- définition des schémas
- indécidabilité
- définition de trois algorithmes :
  - STAB : extension des tableaux sémantiques aux schémas
    $\longrightarrow$ pb : peu adaptée aux schémas imbriqués
  - DPLL* : extension de DPLL aux schémas
    $\longrightarrow$ définie dans le but de gérer les schémas imbriqués
  - SchAut : spécialisation de STAB aux schémas réguliers
- étude de la notion de cycle, indépendamment du calcul
- décidabilité pour deux classes de schémas
  (schémas réguliers et régulièrement imbriqués)
- complexité pour les schémas réguliers (borne sup. uniquement)
- implémentation de SchAut ($\longrightarrow$ « RegSTAB »)

RegSTAB

- outil en ligne de commande `regstab`
- entrée = schéma régulier
  - retourne `UNSATISFIABLE` si schéma insatisfaisable
  - retourne `SATISFIABLE` si schéma satisfaisable
  - **termine tout le temps**
- implémenté en Ocaml
- `http ://regstab.forge.ocamlcore.org/`
- Exemple :

```
varavantinos:~/RegSTAB/bin>./regstab
Taking input from stdin.
P_1 /\ /\i=1..n (P_i -> P_i+1) /\ ~P_n+1 | n>=0
UNSATISFIABLE

varavantinos:~/RegSTAB/bin>./regstab
P_1 /\ /\i=1..n (P_i -> P_i+1) /\ P_n+1 | n>=0
SATISFIABLE (for n = 0)
```

- comparaisons avec ACL2 (successeur de Nqthm) : seul un exemple trivial termine.

Travaux futurs

- approfondissement des liens avec des travaux connexes :
  - logiques temporelles (LTL en particulier)
  - combinaison de théories
- implémentation :
  - DPLL*
  - intégration à des assistants de preuve (p.ex. Coq)
- extensions :
  - application de la notion de cycle au raisonnement inductif en général
  - **schémas de formules du premier ordre**

Merci !

Allons boire des coups !