# Deduction contrainte

Jean-Pierre Jouannaud
École Polytechnique
91400 Palaiseau, France

email: jouannaud@lix.polytechnique.fr
http: //w$^3$.lix.polytechnique.fr/Labo/Jean-Pierre.Jouannaud

Project LogiCal, Pôle Commun de Recherche en Informatique du Plateau de Saclay, CNRS, École Polytechnique, INRIA, Université Paris-Sud.

November 22, 2005

# Outline

# Historique et Motivations

# Historique de la programmation contrainte

- Programmation logique contrainte en PROLOG [A. Colmerauer]
- Programmation logique contrainte en PROLOG II [A. Colmerauer, 82]
- Programmation contrainte en CHIP [M. Dincbas, 87]
- Programmation contrainte paramètrée CLP(X) [J.-L. Lassez, M. Maher, 88]
- Programmation concurrente contrainte [V. Sarasvat, 90]
- Contraintes de typage sûr [F. Pottier, 95]

# Historique de la déduction contrainte

- Récriture ordo-sortée [Goguen, Jouannaud, Meseguer, 85]
- Déduction contrainte [A. Degtyarev, A. Voronkov, 86 ; C.&H. Kirchner, M. Rusinowitch, 90]
- Complétion ordo-sortée [H. Comon, 91]
- Complétion basique [R. Nieuwenhuis, A. Rubio, 92]
- Stratégies basiques ordonnées de résolution [H. Ganzinger et al., 94]
- Model-checking modulo [A. Podelski, 95]
- SAT modulo theories [R. Nieuwenhuis et al.]

Programmation logique pure contrainte :

$$\frac{A \longleftarrow C \parallel \phi \qquad \longleftarrow A', C' \parallel \phi'}{\longleftarrow C, C' \parallel \phi \wedge \phi' \wedge A \doteq A'}$$

Éliminer $\longleftarrow C \parallel \phi$ si $\phi$ est insatisfiable

$$\frac{\longleftarrow \parallel \phi}{\text{Retourner } \mathit{Sol}(\phi) \text{ si } \phi \text{ est satisfiable}}$$

# Ingédients

- les contraintes sont des conjonctions d'égalités appellées contraintes d'unification
- les substitutions ne sont pas *appliquées* mais accumulées par conjonction logique : on les dira *héritées*.
- nécessité d'un algorithme de décision du vide d'une accumulation de contraintes
- nécessité d'un algorithme de résolution d'une accumulation de contraintes
- Solutions retournées sous forme d'une contrainte

# Ingédients

- les contraintes sont des conjonctions d'égalités appellées contraintes d'unification
- les substitutions ne sont pas *appliquées* mais accumulées par conjonction logique : on les dira *héritées*.
- nécessité d'un algorithme de décision du vide d'une accumulation de contraintes
- nécessité d'un algorithme de résolution d'une accumulation de contraintes
- Solutions retournées sous forme d'une contrainte

# Ingédients

- les contraintes sont des conjonctions d'égalités appellées contraintes d'unification
- les substitutions ne sont pas *appliquées* mais accumulées par conjonction logique : on les dira *héritées*.
- nécessité d'un algorithme de décision du vide d'une accumulation de contraintes
- nécessité d'un algorithme de résolution d'une accumulation de contraintes
- Solutions retournées sous forme d'une contrainte

# Ingédients

- les contraintes sont des conjonctions d'égalités appellées contraintes d'unification
- les substitutions ne sont pas *appliquées* mais accumulées par conjonction logique : on les dira *héritées*.
- nécessité d'un algorithme de décision du vide d'une accumulation de contraintes
- nécessité d'un algorithme de résolution d'une accumulation de contraintes
- Solutions retournées sous forme d'une contrainte

# Ingédients

- les contraintes sont des conjonctions d'égalités appellées contraintes d'unification
- les substitutions ne sont pas *appliquées* mais accumulées par conjonction logique : on les dira *héritées*.
- nécessité d'un algorithme de décision du vide d'une accumulation de contraintes
- nécessité d'un algorithme de résolution d'une accumulation de contraintes
- Solutions retournées sous forme d'une contrainte

# PROLOG II

- Changement du domaine d'interprétation, le domaine des arbres infinis.
- Dans ce domaine, l'équation $f(f(x)) \doteq f(x)$ a pour unificateur principal l'arbre infini point fixe de l'équation $x \doteq f(x)$.
- Changement du langage de contraintes avec introduction du prédicat $\neq$ et du quantificateur existentiel pour l'éliminer.
- Avec une signature où 0 et *succ* sont les deux seuls symboles, la contrainte $x \neq 0$ a pour solution la contrainte $\exists y\ x \doteq S(y)$.
- La contrainte $x \neq z$ est en forme normale : l'ensemble des solutions ne peut plus être décrit par un ensemble fini de substitutions.

# PROLOG II

- Changement du domaine d'interprétation, le domaine des arbres infinis.
- Dans ce domaine, l'équation $f(f(x)) \doteq f(x)$ a pour unificateur principal l'arbre infini point fixe de l'équation $x \doteq f(x)$.
- Changement du langage de contraintes avec introduction du prédicat $\not\doteq$ et du quantificateur existentiel pour l'éliminer.
- Avec une signature où 0 et *succ* sont les deux seuls symboles, la contrainte $x \not\doteq 0$ a pour solution la contrainte $\exists y\ x \doteq S(y)$.
- La contrainte $x \not\doteq z$ est en forme normale : l'ensemble des solutions ne peut plus être décrit par un ensemble fini de substitutions.

# PROLOG II

- Changement du domaine d'interprétation, le domaine des arbres infinis.
- Dans ce domaine, l'équation $f(f(x)) \doteq f(x)$ a pour unificateur principal l'arbre infini point fixe de l'équation $x \doteq f(x)$.
- Changement du langage de contraintes avec introduction du prédicat $\neq$ et du quantificateur existentiel pour l'éliminer.
- Avec une signature où 0 et *succ* sont les deux seuls symboles, la contrainte $x \neq 0$ a pour solution la contrainte $\exists y\ x \doteq S(y)$.
- La contrainte $x \neq z$ est en forme normale : l'ensemble des solutions ne peut plus être décrit par un ensemble fini de substitutions.

# PROLOG II

- Changement du domaine d'interprétation, le domaine des arbres infinis.
- Dans ce domaine, l'équation $f(f(x)) \doteq f(x)$ a pour unificateur principal l'arbre infini point fixe de l'équation $x \doteq f(x)$.
- Changement du langage de contraintes avec introduction du prédicat $\not\doteq$ et du quantificateur existentiel pour l'éliminer.
- Avec une signature où 0 et *succ* sont les deux seuls symboles, la contrainte $x \not\doteq 0$ a pour solution la contrainte $\exists y\ x \doteq S(y)$.
- La contrainte $x \not\doteq z$ est en forme normale : l'ensemble des solutions ne peut plus être décrit par un ensemble fini de substitutions.

# PROLOG II

- Changement du domaine d'interprétation, le domaine des arbres infinis.
- Dans ce domaine, l'équation $f(f(x)) \doteq f(x)$ a pour unificateur principal l'arbre infini point fixe de l'équation $x \doteq f(x)$.
- Changement du langage de contraintes avec introduction du prédicat $\not\doteq$ et du quantificateur existentiel pour l'éliminer.
- Avec une signature où 0 et *succ* sont les deux seuls symboles, la contrainte $x \not\doteq 0$ a pour solution la contrainte $\exists y\ x \doteq S(y)$.
- La contrainte $x \not\doteq z$ est en forme normale : l'ensemble des solutions ne peut plus être décrit par un ensemble fini de substitutions.

# CHIP

- Nouveau changement du domaine d'interprétation : les domaines finis comme $\{bleu, blanc, rouge\}$.
- Nouveau changement du langage de contraintes avec l' introduction de prédicats utiles en pratique pour l'expression de problèmes dans des domaines d'application variés, comme le prédicat $Alldifferent(\overline{x})$.
- Applications aux problèmes d'allocation, de planification, d'ordonnancement, etc.
- On peut bien sûr prendre d'autres domaines d'interprétation, comme les réels pour les calculs financiers.

# CHIP

- Nouveau changement du domaine d'interprétation : les domaines finis comme $\{bleu, blanc, rouge\}$.
- Nouveau changement du langage de contraintes avec l' introduction de prédicats utiles en pratique pour l'expression de problèmes dans des domaines d'application variés, comme le prédicat $Alldifferent(\overline{x})$.
- Applications aux problèmes d'allocation, de planification, d'ordonnancement, etc.
- On peut bien sûr prendre d'autres domaines d'interprétation, comme les réels pour les calculs financiers.

# CHIP

- Nouveau changement du domaine d'interprétation : les domaines finis comme $\{bleu, blanc, rouge\}$.
- Nouveau changement du langage de contraintes avec l' introduction de prédicats utiles en pratique pour l'expression de problèmes dans des domaines d'application variés, comme le prédicat $Alldifferent(\overline{x})$.
- Applications aux problèmes d'allocation, de planification, d'ordonnancement, etc.
- On peut bien sûr prendre d'autres domaines d'interprétation, comme les réels pour les calculs financiers.

# CHIP

- Nouveau changement du domaine d'interprétation : les domaines finis comme $\{bleu, blanc, rouge\}$.
- Nouveau changement du langage de contraintes avec l' introduction de prédicats utiles en pratique pour l'expression de problèmes dans des domaines d'application variés, comme le prédicat $Alldifferent(\overline{x})$.
- Applications aux problèmes d'allocation, de planification, d'ordonnancement, etc.
- On peut bien sûr prendre d'autres domaines d'interprétation, comme les réels pour les calculs financiers.

# Déduction contrainte

Résolution et Factorisation contraintes pour des clauses quelconques :

$$\frac{+A \vee C \parallel \phi \quad -A' \vee C' \parallel \phi'}{C \vee C' \parallel \phi \wedge \phi' \wedge A \doteq A'}$$

$$\frac{+A \vee +A' \vee C \parallel \phi}{+A \vee C \parallel \phi \wedge A \doteq A'}$$

Éliminer $C \parallel \phi$ si $\phi$ est insatisfiable

$$\frac{\square \parallel \phi}{\text{Retourner } \textit{Succès} \text{ si } \phi \text{ est satisfiable}}$$

# Déduction contrainte, suite

Résolution et Factorisation ordonnées contraintes :

$$\frac{+A \vee C \parallel \phi \quad -A' \vee C' \parallel \phi'}{C \vee C' \parallel \phi \wedge \phi' \wedge A \doteq A' \wedge A > C \wedge A' > C'}$$

$$\frac{+A \vee +A' \vee C \parallel \phi}{+A \vee C \parallel \phi \wedge A \doteq A' \wedge A > C}$$

Éliminer $C \parallel \phi$ si $\phi$ est insatisfiable

$$\frac{\square \parallel \phi}{\text{Retourner } \textit{Succès} \text{ si } \phi \text{ est satisfiable}}$$

# Déduction contrainte, suite

- Ces règles d'inférence sont obtenues à partir du système d'inférence de résolution et factorisation ordonnées en transformant les conditions d'application en contraintes.
- La complétude des règles obtenues s'obtient à partir de celle du système d'origine par simulation des dérivations de la clause vide dans l'ancien calcul par le nouveau.
- L'héritage des contraintes au cours des inférences utilise un *test* de satisfiabilité qui économise le calcul des unificateurs.
- Les inférences ordonnées approximent la comparaison $\forall\gamma\ A\gamma > C\gamma$ avec la condition $C \not\geq A$ qui est plus prolifique.

# Déduction contrainte, suite

- Ces règles d'inférence sont obtenues à partir du système d'inférence de résolution et factorisation ordonnées en transformant les conditions d'application en contraintes.
- La complétude des règles obtenues s'obtient à partir de celle du système d'origine par simulation des dérivations de la clause vide dans l'ancien calcul par le nouveau.
- L'héritage des contraintes au cours des inférences utilise un *test* de satisfiabilité qui économise le calcul des unificateurs.
- Les inférences ordonnées approximent la comparaison $\forall\gamma\ A\gamma > C\gamma$ avec la condition $C \not\geq A$ qui est plus prolifique.

# Déduction contrainte, suite

- Ces règles d'inférence sont obtenues à partir du système d'inférence de résolution et factorisation ordonnées en transformant les conditions d'application en contraintes.
- La complétude des règles obtenues s'obtient à partir de celle du système d'origine par simulation des dérivations de la clause vide dans l'ancien calcul par le nouveau.
- L'héritage des contraintes au cours des inférences utilise un *test* de satisfiabilité qui économise le calcul des unificateurs.
- Les inférences ordonnées approximent la comparaison $\forall\gamma\ A\gamma > C\gamma$ avec la condition $C \not\geq A$ qui est plus prolifique.

# Déduction contrainte, suite

- Ces règles d'inférence sont obtenues à partir du système d'inférence de résolution et factorisation ordonnées en transformant les conditions d'application en contraintes.
- La complétude des règles obtenues s'obtient à partir de celle du système d'origine par simulation des dérivations de la clause vide dans l'ancien calcul par le nouveau.
- L'héritage des contraintes au cours des inférences utilise un *test* de satisfiabilité qui économise le calcul des unificateurs.
- Les inférences ordonnées approximent la comparaison $\forall\gamma\ A\gamma > C\gamma$ avec la condition $C \not\geq A$ qui est plus prolifique.

# Déduction contrainte, suite

- les contraintes sont *symboliques*, c-à-d interprétées dans un domaine de Herbrand.
- Le vocabulaire qui les exprime dépend du problème à modéliser. Il pourra être formé de contraintes d'égalités, d'ordre, d'appartenance à des langages rationnels d'arbres, ou d'une combinaison d'icelles.
- Le langage de contraintes dépend également de l'algorithme de résolution de contraintes. Il sera clos par conjonction, et par renommage des variables liées s'il y en a. Il sera souvent clos par quantification existentielle, et parfois par négation. Il pourra comprendre tout le premier ordre.

# Déduction contrainte, suite

- les contraintes sont *symboliques*, c-à-d interprétées dans un domaine de Herbrand.
- Le vocabulaire qui les exprime dépend du problème à modéliser. Il pourra être formé de contraintes d'égalités, d'ordre, d'appartenance à des langages rationnels d'arbres, ou d'une combinaison d'icelles.
- Le langage de contraintes dépend également de l'algorithme de résolution de contraintes. Il sera clos par conjonction, et par renommage des variables liées s'il y en a. Il sera souvent clos par quantification existentielle, et parfois par négation. Il pourra comprendre tout le premier ordre.

# Déduction contrainte, suite

- les contraintes sont *symboliques*, c-à-d interprétées dans un domaine de Herbrand.
- Le vocabulaire qui les exprime dépend du problème à modéliser. Il pourra être formé de contraintes d'égalités, d'ordre, d'appartenance à des langages rationnels d'arbres, ou d'une combinaison d'icelles.
- Le langage de contraintes dépend également de l'algorithme de résolution de contraintes. Il sera clos par conjonction, et par renommage des variables liées s'il y en a. Il sera souvent clos par quantification existentielle, et parfois par négation. Il pourra comprendre tout le premier ordre.

# Récriture contrainte

- Équation contrainte : $l = r \parallel \phi$
  où $\phi$ est la contrainte portant sur les variables de l'équation $l = r$,
  le symbole = étant considéré commutatif.
- Récriture contrainte :

$$s \xrightarrow[l=r \parallel \phi]{p} s[r\sigma]_p \quad \text{si} \quad \begin{cases} s|_p = l\sigma \\ l\sigma \succ r\sigma \\ \phi\sigma \text{ est vraie} \end{cases}$$

- Règle contrainte :
  $l \to r \parallel \phi$ si $l\phi \succeq r\phi$
  $l \to r \parallel \phi \wedge l \succ r$
  en internalisant le test $l\sigma \succ r\sigma$.

# Récriture contrainte

- Équation contrainte : $l = r \parallel \phi$
  où $\phi$ est la contrainte portant sur les variables de l'équation $l = r$,
  le symbole = étant considéré commutatif.
- Récriture contrainte :

$$s \xrightarrow[l=r \parallel \phi]{p} s[r\sigma]_p \quad \text{si} \quad \begin{cases} s|_p = l\sigma \\ l\sigma \succ r\sigma \\ \phi\sigma \text{ est vraie} \end{cases}$$

- Règle contrainte :
  $l \to r \parallel \phi$ si $l\phi \succeq r\phi$
  $l \to r \parallel \phi \wedge l \succ r$
  en internalisant le test $l\sigma \succ r\sigma$.

# Récriture contrainte

- Équation contrainte : $l = r \parallel \phi$
  où $\phi$ est la contrainte portant sur les variables de l'équation $l = r$,
  le symbole = étant considéré commutatif.
- Récriture contrainte :

$$s \xrightarrow[l=r \parallel \phi]{p} s[r\sigma]_p \quad \text{si} \quad \begin{cases} s|_p = l\sigma \\ l\sigma \succ r\sigma \\ \phi\sigma \text{ est vraie} \end{cases}$$

- Règle contrainte :
  $l \to r \parallel \phi$ si $l\phi \succeq r\phi$
  $l \to r \parallel \phi \wedge l \succ r$
  en internalisant le test $l\sigma \succ r\sigma$.

# Stratégies de superposition

- Complétion basique :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p = l \parallel \phi \wedge \psi \wedge g|_p \doteq l}$$

  Pour une règle, la superposition peut être restreinte à son membre gauche.

- Les superpositions ont lieu dans les règles, et pas dans les substitutions issues d'inférences antérieures.
- Complétion basique ordonnée :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p \doteq d \parallel \phi \wedge \psi \wedge g|_p \doteq l \wedge g \succ d \wedge l \succ r}$$

- Ces règles sont-elles complètes ?

# Stratégies de superposition

- Complétion basique :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p = l \parallel \phi \wedge \psi \wedge g|_p \doteq l}$$

  Pour une règle, la superposition peut être restreinte à son membre gauche.
- Les superpositions ont lieu dans les règles, et pas dans les substitutions issues d'inférences antérieures.
- Complétion basique ordonnée :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p \doteq d \parallel \phi \wedge \psi \wedge g|_p \doteq l \wedge g \succ d \wedge l \succ r}$$

- Ces règles sont-elles complètes ?

# Stratégies de superposition

- Complétion basique :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p = l \parallel \phi \wedge \psi \wedge g|_p \doteq l}$$

  Pour une règle, la superposition peut être restreinte à son membre gauche.
- Les superpositions ont lieu dans les règles, et pas dans les substitutions issues d'inférences antérieures.
- Complétion basique ordonnée :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p \doteq d \parallel \phi \wedge \psi \wedge g|_p \doteq l \wedge g \succ d \wedge l \succ r}$$

- Ces règles sont-elles complètes ?

# Stratégies de superposition

- Complétion basique :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p = l \parallel \phi \wedge \psi \wedge g|_p \doteq l}$$

  Pour une règle, la superposition peut être restreinte à son membre gauche.
- Les superpositions ont lieu dans les règles, et pas dans les substitutions issues d'inférences antérieures.
- Complétion basique ordonnée :

$$\frac{g = d \parallel \phi \quad l = r \parallel \psi}{g[r]_p \doteq d \parallel \phi \wedge \psi \wedge g|_p \doteq l \wedge g \succ d \wedge l \succ r}$$

- Ces règles sont-elles complètes ?

# Récriture ordo-sortée [OBJ]

| | |
|---|---|
| sort | s, t |
| subsorts | $s < t$ |
| op | a : s |
| op | b, c : t |
| op | f : t $\longrightarrow$ t |
| var | x : s |
| rule | f(x) $\longrightarrow$ b |
| rule | a $\longrightarrow$ c |

Les deux expressions $b$ et $f(c)$ sont syntaxiquement valides et différentes, mais $b \longleftarrow f(a) \longrightarrow f(c)$.

Une paire critique cachée tue la confluence.

Quelle est la notion de règle contrainte ?

# Récriture ordo-sortée [OBJ]

| | |
|---|---|
| sort | s, t |
| subsorts | $s < t$ |
| op | a : s |
| op | b, c : t |
| op | f : t $\longrightarrow$ t |
| var | x : s |
| rule | f(x) $\longrightarrow$ b |
| rule | a $\longrightarrow$ c |

Les deux expressions $b$ et $f(c)$ sont syntaxiquement valides et différentes, mais $b \longleftarrow f(a) \longrightarrow f(c)$.

Une paire critique cachée tue la confluence.

Quelle est la notion de règle contrainte ?

# Logique et Contraintes

# Système de contraintes

- Vocabulaire $L_C$ formé d'un ensemble $\mathcal{F}_C$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}_C$ de symboles de relations.
- Un sous-ensemble $\mathcal{C}_C$ des formules du premier ordre sur $L_C$ contenant les formules atomiques, clos par $\wedge$, $\exists$ et renommage des variables liées.
- Une structure du premier ordre $\mathcal{A}_C$ qui interprète les formules sur $L_C$. On notera par $[\![\phi]\!]$ l'ensemble des substitutions closes $\gamma$ vérifiant $\mathcal{A} \models \phi\gamma$, appellées *solutions* de $\phi$.
- Un algorithme de décision de la satisfiabilité (ou non-vacuité de l'ensemble des solutions) des formules de $\mathcal{C}_C$.

# Système de contraintes

- Vocabulaire $L_C$ formé d'un ensemble $\mathcal{F}_C$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}_C$ de symboles de relations.
- Un sous-ensemble $\mathcal{C}_C$ des formules du premier ordre sur $L_C$ contenant les formules atomiques, clos par $\wedge$, $\exists$ et renommage des variables liées.
- Une structure du premier ordre $\mathcal{A}_C$ qui interprète les formules sur $L_C$. On notera par $[\![\phi]\!]$ l'ensemble des substitutions closes $\gamma$ vérifiant $\mathcal{A} \models \phi\gamma$, appellées *solutions* de $\phi$.
- Un algorithme de décision de la satisfiabilité (ou non-vacuité de l'ensemble des solutions) des formules de $\mathcal{C}_C$.

# Système de contraintes

- Vocabulaire $L_C$ formé d'un ensemble $\mathcal{F}_C$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}_C$ de symboles de relations.
- Un sous-ensemble $\mathcal{C}_C$ des formules du premier ordre sur $L_C$ contenant les formules atomiques, clos par $\wedge$, $\exists$ et renommage des variables liées.
- Une structure du premier ordre $\mathcal{A}_C$ qui interprète les formules sur $L_C$. On notera par $[\![\phi]\!]$ l'ensemble des substitutions closes $\gamma$ vérifiant $\mathcal{A} \models \phi\gamma$, appellées *solutions* de $\phi$.
- Un algorithme de décision de la satisfiabilité (ou non-vacuité de l'ensemble des solutions) des formules de $\mathcal{C}_C$.

# Système de contraintes

- Vocabulaire $L_C$ formé d'un ensemble $\mathcal{F}_C$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}_C$ de symboles de relations.
- Un sous-ensemble $\mathcal{C}_C$ des formules du premier ordre sur $L_C$ contenant les formules atomiques, clos par $\wedge$, $\exists$ et renommage des variables liées.
- Une structure du premier ordre $\mathcal{A}_C$ qui interprète les formules sur $L_C$. On notera par $[\![\phi]\!]$ l'ensemble des substitutions closes $\gamma$ vérifiant $\mathcal{A} \models \phi\gamma$, appellées *solutions* de $\phi$.
- Un algorithme de décision de la satisfiabilité (ou non-vacuité de l'ensemble des solutions) des formules de $\mathcal{C}_C$.

# Formules contraintes

- Vocabulaire $L$ formé d'un ensemble $\mathcal{F}$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}$ de symboles de relations.
- Les clauses contraintes sont de la forme $C \parallel \phi$, où $C$ est une clause sur le langage $L$, et $\phi$ une contrainte du langage $\mathcal{C}_C$.
- Une clause contrainte $C \parallel \phi$ dénote l'ensemble des clauses closes $C\gamma$ appellées *instances* de $C \parallel \phi$, pour toute substitution close $\gamma$ solution de $\phi$.
- La clause $C \parallel \phi \wedge \psi$ dénote donc l'ensemble des clauses closes $C\gamma$, où $\gamma \in [\![\phi]\!] \cap [\![\psi]\!]$.

# Formules contraintes

- Vocabulaire $L$ formé d'un ensemble $\mathcal{F}$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}$ de symboles de relations.
- Les clauses contraintes sont de la forme $C \parallel \phi$, où $C$ est une clause sur le langage $L$, et $\phi$ une contrainte du langage $\mathcal{C}_C$.
- Une clause contrainte $C \parallel \phi$ dénote l'ensemble des clauses closes $C\gamma$ appellées *instances* de $C \parallel \phi$, pour toute substitution close $\gamma$ solution de $\phi$.
- La clause $C \parallel \phi \wedge \psi$ dénote donc l'ensemble des clauses closes $C\gamma$, où $\gamma \in [\![\phi]\!] \cap [\![\psi]\!]$.

# Formules contraintes

- Vocabulaire $L$ formé d'un ensemble $\mathcal{F}$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}$ de symboles de relations.
- Les clauses contraintes sont de la forme $C \parallel \phi$, où $C$ est une clause sur le langage $L$, et $\phi$ une contrainte du langage $\mathcal{C}_C$.
- Une clause contrainte $C \parallel \phi$ dénote l'ensemble des clauses closes $C\gamma$ appellées *instances* de $C \parallel \phi$, pour toute substitution close $\gamma$ solution de $\phi$.
- La clause $C \parallel \phi \wedge \psi$ dénote donc l'ensemble des clauses closes $C\gamma$, où $\gamma \in [\![\phi]\!] \cap [\![\psi]\!]$.

# Formules contraintes

- Vocabulaire $L$ formé d'un ensemble $\mathcal{F}$ de symboles de fonction, $\mathcal{X}$ de symboles de variables, et $\mathcal{R}$ de symboles de relations.
- Les clauses contraintes sont de la forme $C \parallel \phi$, où $C$ est une clause sur le langage $L$, et $\phi$ une contrainte du langage $\mathcal{C}_C$.
- Une clause contrainte $C \parallel \phi$ dénote l'ensemble des clauses closes $C\gamma$ appellées *instances* de $C \parallel \phi$, pour toute substitution close $\gamma$ solution de $\phi$.
- La clause $C \parallel \phi \wedge \psi$ dénote donc l'ensemble des clauses closes $C\gamma$, où $\gamma \in [\![\phi]\!] \cap [\![\psi]\!]$.

# Complétion basique

# Principe

- L'objectif est que toute preuve avec les équations de départ devienne une preuve par récriture avec les équations inférées.
- On donne un premier système d'inférence avec des règles de simplification simples.
- On utilise un ordre de réduction $\succ$ total sur les termes clos, et on note $l \longrightarrow r$ toute équation $l = r$ telle que $l \succ r$.
- On montre la complétude de la complétion basique en utilisant la technique de récriture de preuves.

# Règles de complétion basique

Cri-
tical
Pair

$$\frac{l = r \parallel \phi \qquad g = d \parallel \psi}{l[d]_p = r \parallel \phi \wedge \psi \wedge l|_p \doteq g}$$

si $\begin{cases} l|_p \notin \mathcal{V}ar(l) \\ \exists \sigma \in Sol(\phi \wedge \psi \wedge l|_p \doteq g) \text{ t.q.} \\ \quad l\sigma \succ r\sigma \text{ et } g\sigma \succ d\sigma \end{cases}$

==========

Trivial $l = r \parallel \phi \rightarrow nil$ si $l\sigma = r\sigma \quad \forall \sigma \models \phi$

Clean $l = r \parallel \phi \wedge x \doteq t \rightarrow l = r \parallel \phi$ si $x \notin \mathcal{V}ar(l, r, \phi)$

# Lemme des paires critiques

- Le lemme des paires critiques n'est pas valide pour des règles contraintes, même dans le cas de contraintes d'unification:

$$\begin{array}{lcll} f(x) & \to & a & \| \; x \doteq b(y) \\ b(c) & \to & c & \| \end{array}$$

La contrainte cache une paire critique.

- Il peut même devenir faux par complétion:

$$\begin{array}{lcll} f(x) & \to & a & \| \; x \doteq b(y) \\ d & \to & b(c) & \| \\ d & \to & c & \| \end{array}$$

L'ajout de la règle $b(c) \to c \;\|$ crée la paire critique précédente qui était cachée.

# Lemme des paires critiques

- Le lemme des paires critiques n'est pas valide pour des règles contraintes, même dans le cas de contraintes d'unification:

$$\begin{array}{lcll} f(x) & \to & a & \| \quad x \doteq b(y) \\ b(c) & \to & c & \| \end{array}$$

  La contrainte cache une paire critique.
- Il peut même devenir faux par complétion:

$$\begin{array}{lcll} f(x) & \to & a & \| \quad x \doteq b(y) \\ d & \to & b(c) & \| \\ d & \to & c & \| \end{array}$$

  L'ajout de la règle $b(c) \to c \ \|$ crée la paire critique précédente qui était cachée.

# Solutions

- Il y a deux solutions à ce problème.
- La première est due à Hubert Comon: on rajoute des variables de contexte lors de l'unification, qui devient de second ordre, reste décidable dans le fragment considéré, mais de complexité non polynomiale.
- La seconde, que nous allons étudier, trouve son origine dans les travaux de Jean-Marie Hullot sur la surréduction : on suppose que les équations de départ sont non contraintes. Les contraintes ont alors pour seul but l'expression élégante d'une restriction de la complétion.

# Solutions

- Il y a deux solutions à ce problème.
- La première est due à Hubert Comon: on rajoute des variables de contexte lors de l'unification, qui devient de second ordre, reste décidable dans le fragment considéré, mais de complexité non polynomiale.
- La seconde, que nous allons étudier, trouve son origine dans les travaux de Jean-Marie Hullot sur la surréduction : on suppose que les équations de départ sont non contraintes. Les contraintes ont alors pour seul but l'expression élégante d'une restriction de la complétion.

# Solutions

- Il y a deux solutions à ce problème.
- La première est due à Hubert Comon: on rajoute des variables de contexte lors de l'unification, qui devient de second ordre, reste décidable dans le fragment considéré, mais de complexité non polynomiale.
- La seconde, que nous allons étudier, trouve son origine dans les travaux de Jean-Marie Hullot sur la surréduction : on suppose que les équations de départ sont non contraintes. Les contraintes ont alors pour seul but l'expression élégante d'une restriction de la complétion.

# Résultat d'une complétion basique

Nous allons dorénavent mémoriser l'étape $i$ de la complétion à laquelle une équation est engendrée en l'écrivant $u =^i v \parallel \phi$, avec un symbole d'égalité décoré par un entier, sans que sa signification en soit changée.

Soit $\{E_i = \{l_k =^j r_k \parallel \phi_k \mid j \leq i\}_k\}_{i \geq 0}$ la suite des ensembles d'équations présentes à l'étape $i$ engendrée à partir de l'ensemble d'équations initial fini $E_0 = \{l_k =^0 r_k \parallel \phi_k\}_k$. On définit les ensembles

$$E^* = \bigcup_i E_i \quad E^\infty = \bigcup_i \bigcap_{j \geq i} E_i$$

ce dernier étant le résultat de la complétion.

# Construction de $R_{E^\infty}$

- On définit le système de récriture clos $R_{E^\infty}$ comme un sous-ensemble $R_E$ d'instances closes $l\gamma =^i r\gamma$ de $E^\infty$ tel que
  - (i) $R_E$ est maximal
  - (ii) $l\gamma \succ r\gamma$
  - (iii) $l\gamma$ est irréductible par $R_E \setminus \{l\gamma \to r\gamma\}$
  - (iv) $\gamma$ est irréductible par $R_E$.
- $R_{E^\infty}$ est un système clos confluent qui termine:
  $R_{E^\infty}$ est localement confluent et on applique le théorème de Newman.
- On note $s\downarrow$ pour $s\downarrow_{R_{E^\infty}}$

# Construction de $R_{E^\infty}$

- On définit le système de récriture clos $R_{E^\infty}$ comme un sous-ensemble $R_E$ d'instances closes $l\gamma =^i r\gamma$ de $E^\infty$ tel que
  - (i) $R_E$ est maximal
  - (ii) $l\gamma \succ r\gamma$
  - (iii) $l\gamma$ est irréductible par $R_E \setminus \{l\gamma \to r\gamma\}$
  - (iv) $\gamma$ est irréductible par $R_E$.
- $R_{E^\infty}$ est un système clos confluent qui termine:
  $R_{E^\infty}$ est localement confluent et on applique le théorème de Newman.
- On note $s{\downarrow}$ pour $s{\downarrow_{R_{E^\infty}}}$

# Construction de $R_{E^\infty}$

- On définit le système de récriture clos $R_{E^\infty}$ comme un sous-ensemble $R_E$ d'instances closes $l\gamma =^i r\gamma$ de $E^\infty$ tel que
  (i) $R_E$ est maximal
  (ii) $l\gamma \succ r\gamma$
  (iii) $l\gamma$ est irréductible par $R_E \setminus \{l\gamma \to r\gamma\}$
  (iv) $\gamma$ est irréductible par $R_E$.
- $R_{E^\infty}$ est un système clos confluent qui termine:
  $R_{E^\infty}$ est localement confluent et on applique le théorème de Newman.
- On note $s\downarrow$ pour $s\downarrow_{R_{E^\infty}}$

# Algèbre des preuves

Étant donné l'ensemble d'équations décorées contraintes $E^*$, on définit les preuves comme des concaténations de preuves élémentaires :

$$u \xrightarrow[s =^i t \,\|\, \phi \in E^*,\, \gamma]{p} v \quad \text{si} \quad s\gamma \succ t\gamma$$

$$u \xleftarrow[s =^i t \,\|\, \phi \in E^*,\, \gamma]{p} v \quad \text{si} \quad t\gamma \succ s\gamma$$

$$u \xleftrightarrow[s =^i t \,\|\, \phi \in E^*,\, \gamma]{p} v \quad \text{si} \quad \begin{cases} s\gamma = t\gamma \text{ ou} \\ s\gamma \succ t\gamma \text{ ou} \\ t\gamma \succ s\gamma \end{cases}$$

La dernière preuve est une notation pratique.

# Récritures de preuves

$$s \underset{R_E\infty}{\overset{p}{\longleftarrow}} u \underset{R_E\infty}{\overset{q}{\longrightarrow}} t \quad \Rightarrow \quad s \underset{R_E\infty}{\longrightarrow} v \underset{R_E\infty}{\longleftarrow} t$$

$$s \underset{E^*}{\longleftrightarrow} s \quad \Rightarrow \quad nil$$

$$s \underset{I=^i r \parallel \phi \wedge x = u \in E^*}{\longleftrightarrow} t \quad \Rightarrow \quad s \underset{I=^{i+1} r \parallel \phi \in E^*}{\longleftrightarrow} t$$

# Récritures de preuves, suite

$$s[l\gamma]_q \overset{q}{\longleftrightarrow} \left\{ \begin{array}{c} l = r \parallel \phi \in E^\infty \\ \gamma = \gamma\downarrow \\ l\gamma = r\gamma \notin R_{E^\infty} \end{array} \right\} s[r\gamma]_q \Rightarrow$$

$$s[l\gamma[g\gamma]_p]_q \underset{g=d \parallel \psi}{\overset{q\cdot p}{\longrightarrow}} s[l\gamma[d\gamma]_p]_q \underset{l[d]_p=r \parallel \phi\wedge\psi\wedge l|_p \doteq g}{\overset{q}{\longleftrightarrow}} s[r\gamma]_q$$

où $\quad l\gamma[g\gamma]_p \underset{g\gamma\to d\gamma\in R_{E^\infty}}{\overset{p}{\longrightarrow}} l\gamma[d\gamma]_p$

# Récritures de preuves, suite

$$s[(l[d]_p)\gamma] \underset{l[d]_p = ^{i+1} r \ \|\ \phi\wedge\psi\wedge l|_p \doteq g \in E^\infty,\ \gamma \neq \gamma\downarrow}{\longleftrightarrow} s[r\gamma] \Rightarrow$$

$$s[(l[d]_p)\gamma] \underset{E_i}{\longleftrightarrow} s[(l[g]_p)\gamma] = s[l\gamma] \underset{E_i}{\longleftrightarrow} s[r\gamma]$$

$$s[l\gamma] \underset{\left\{ \begin{array}{c} l = r \ \|\ \phi \in E_0 \\ \gamma \neq \gamma\downarrow \end{array} \right\}}{\longleftrightarrow} s[r\gamma] \Rightarrow$$

$$s[l\gamma] \underset{R_{E^\infty}}{\overset{+}{\longrightarrow}} s[l\gamma\downarrow] \underset{\left\{ \begin{array}{c} l = r \ \|\ \phi \in E_0 \\ \gamma\downarrow \end{array} \right\}}{\longleftrightarrow} s[r\gamma\downarrow] \underset{R_{E^\infty}}{\overset{+}{\longleftarrow}} s[r\gamma]$$

# Équité

- Une séquence de complétion $\{E_i\}_i$ est *équitable* si toute paire critique entre équations de $E_i$ est calculée à une étape $j > i$.
- Un ensemble d'équations contraintes est sans superposition dans les contraintes si les contraintes n'ont aucun symbole de fonction commun avec les têtes des équations. Cette propriété est satisfaite dans le cas d'équations non contraintes.

# Équité

- Une séquence de complétion $\{E_i\}_i$ est *équitable* si toute paire critique entre équations de $E_i$ est calculée à une étape $j > i$.
- Un ensemble d'équations contraintes est sans superposition dans les contraintes si les contraintes n'ont aucun symbole de fonction commun avec les têtes des équations. Cette propriété est satisfaite dans le cas d'équations non contraintes.

# Formes normales de preuves

## Lemma

*Étant donnés un ensemble d'équations contraintes $E_0$ sans superposition dans les contraintes, et une séquence de complétion équitable $\{E_i\}_i$ issue de $E_0$, les preuves closes avec les équations de $E^*$ qui sont en forme normale pour les règles de preuve sont des preuves par récriture avec les règles de $R^\infty$.*

Ce lemme énonce à la fois la correction des règles de preuve et la réductibilité des preuves formées avec les équations de $E^* \setminus R_{E^\infty}$.

## Proof.

Soit $u[l\gamma]_p \longrightarrow^{p}_{l=r \parallel \phi} u[r\gamma]_p$, avec $l\gamma =^{i} r\gamma \notin R_{E\infty}$. Le cas $l\gamma = r\gamma$ est réductible par la seconde règle. Sinon $l\gamma \succ r\gamma$ et $l\gamma$ ou $\gamma$ sont réductibles par $R_{E\infty}$.

1. $\gamma \neq \gamma\downarrow$ et $l = r \parallel \phi \notin E_0$. La preuve se réduit par la cinquième règle.
2. $\gamma \neq \gamma\downarrow$ et $l = r \parallel \phi \in E_0$. Alors $\gamma = \phi\gamma'$. L'absence de superposition dans les contraintes fait que $\gamma\downarrow = \phi \cdot \gamma'\downarrow$, et donc $\gamma\downarrow$ satisfait $\phi$. La dernière règle s'applique.
3. $\gamma = \gamma\downarrow$. $s\gamma$ est réductible à une position de $s$. Par équité, la quatrième règle s'applique.

# Terminaison

**Lemma**

*La récriture de preuve fait décroître l'ordre sur les preuves.*

**Proof.**

On montre que chaque règle fait décroitre un certain ordre bien-fondé sur les preuves. Un tel ordre est donné ci-après. □

On en déduit :

**Theorem**

*La complétion basique est complète pour les ensembles d'équations qui sont sans superposition dans leurs contraintes.*

# Interprétation d'une étape de preuve atomique

$$[\![ s \underset{l =^{i} r \parallel \phi,\gamma}{\overset{p}{\longleftrightarrow}} t ]\!] =$$

$$
\begin{aligned}
< 0, \bot, \bot, \bot > \quad & \text{si} \quad l\gamma = r\gamma \\
< 0, \bot, \bot, s > \quad & \text{si} \quad l\gamma \rightarrow r\gamma \in R_{E^\infty} \\
< 0, \bot, \bot, t > \quad & \text{si} \quad l\gamma \leftarrow r\gamma \in R_{E^\infty} \\
< 0, \bot, s|_p, \bot > \quad & \text{si} \quad l\gamma = r\gamma \notin R_{E^\infty}, \gamma = \gamma\downarrow, l\gamma \\
< 0, \bot, t|_p, \bot > \quad & \text{si} \quad l\gamma = r\gamma \notin R_{E^\infty}, \gamma = \gamma\downarrow, r\gamma \\
< 0, \mathcal{V}ar(l, r)\gamma, \bot, \bot > \quad & \text{si} \quad l\gamma = r\gamma \notin R_{E^\infty}, \\
& \quad \gamma \neq \gamma\downarrow \text{ et } i = 0 \\
< i, \bot, \bot, \bot > \quad & \text{si} \quad l\gamma = r\gamma \notin R_{E^\infty}, \\
& \quad \gamma \neq \gamma\downarrow \text{ et } i > 0
\end{aligned}
$$

# Ordre sur les interprétation

Une preuve est interprétée par le multiensemble des interprétations de ses étapes atomiques. En particulier, la preuve *nil* est interprétée par le multiensemble vide.

On compare les interprétations avec l'ordre

$$(>_{\mathbf{N}}, \succ_{mul}, \succ, \succ)_{lex}$$

Le lecteur vérifiera la décroissance des règles et pourra améliorer l' ordre pour tenir compte de la règle *clean*.

# Exemple

On utilise l'ordre $rpo(g > f > a > b)$

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | $a(b)$ | $\to$ | $b$ | | | $\in E_0$ |
| 2 | $f(g(x))$ | $\to$ | $g(x)$ | | | $\in E_0$ |
| 3 | $g(a(x))$ | $\to$ | $b$ | | | $\in E_0$ |
| 4 | $g(b)$ | $\to$ | $b$ | $\|$ | $x \doteq b$ | $CP(1,3)$ |
| 5 | $g(b)$ | $\to$ | $b$ | | | $Clean(4)$ |
| 6 | $g(x)$ | $\to$ | $f(b)$ | $\|$ | $x \doteq a(z)$ | $CP(2,3)$ |
| 7 | $g(x)$ | $\to$ | $f(b)$ | $\|$ | $x \doteq b$ | $CP(2,5)$ |
| 8 | $g(x)$ | $\to$ | $f(f(b))$ | $\|$ | $x \doteq a(z)$ | $CP(2,6)$ |
| | | | | | $x \doteq a(z)$ | |
| 9 | $f(b)$ | $\to$ | $b$ | $\|$ | $\wedge$ | $CP(3,6)$ |
| | | | | | $x \doteq a(x')$ | |
| 10 | $f(b)$ | $\to$ | $b$ | | | $Clean(9)$ |

# Exemple, suite

$$
\begin{array}{llcll|ll}
11 & f(b) & \to & b & \| & x \doteq b & CP(5,6) \\
12 & f(b) & \to & b & & & Clean(11) \\
13 & g(x) & \to & f(f(b)) & \| & x \doteq b & CP(2,7) \\
\ldots & & & & & &
\end{array}
$$

La complétion diverge, mais on peut prouver $g(b) = b$ par récriture.

# Complétion basique avec simplification

# Règle de simplification

On définit

$$g = d \parallel \phi \rightarrow g[r\sigma]_p = d \parallel \phi\sigma \text{ si } g \xrightarrow[l=r \parallel \psi]{p} r\sigma$$

qui est le cas particulier de calcul de paire critique obtenu lorsque le membre gauche de l'une des règles est réductible par l'autre. Dans le cas où $g = d \parallel \phi$ est orientable en la règle contrainte $g \rightarrow d \parallel \phi$, on réduira $g$ comme $d$.

# Première stratégie équitable

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | $a(b)$ | $\to$ | $b$ | | | $\in E_0$ |
| 2 | $f(g(x))$ | $\to$ | $g(x)$ | | | $\in E_0$ |
| 3 | $g(a(x))$ | $\to$ | $b$ | | | $\in E_0$ |
| 4 | $g(b)$ | $\to$ | $b$ | $\parallel$ | $x \doteq b$ | $CP(1,3)$ |
| 5 | $g(b)$ | $\to$ | $b$ | | | $Clean(4)$ |
| 6 | $g(x)$ | $\to$ | $f(b)$ | $\parallel$ | $x \doteq a(z)$ | $CP(2,3)$ |
| 7 | $f(b)$ | $\to$ | $b$ | | | $Simpl(3)$ |
| 8 | $g(x)$ | $\to$ | $b$ | $\parallel$ | $x \doteq a(z)$ | $Simpl(6)$ |
| 9 | $g(x)$ | $\to$ | $f(b)$ | $\parallel$ | $x \doteq b$ | $CP(2,5)$ |
| 10 | $g(x)$ | $\to$ | $b$ | $\parallel$ | $x \doteq b$ | $Simpl(9)$ |
| 11 | $b$ | $=$ | $b$ | $Simpl(5)$ puis $Elim(11)$ | | |
| 13 | $g(x)$ | $\to$ | $f(b)$ | $\parallel$ | $x \doteq a(z)$ | $CP(2,8)$ |
| 14 | $f(b)$ | $\to$ | $b$ | $\parallel$ | $x \doteq a(z)$ | $Simpl(13)$ |

# Première stratégie équitable, suite

15 $\quad$ *Simpl*(14) puis *Elim*(15)

16 $\quad g(x) \rightarrow f(b) \parallel x \doteq b \quad CP(2,10)$

Simpl(16), puis Simpl(17) et Elim(18). FIN

Le système obtenu est équivalent au système de règles sans contraintes :

$$
\begin{array}{lrcl}
1 & a(b) & \rightarrow & b \\
2 & f(g(x)) & \rightarrow & g(x) \\
7 & f(b) & \rightarrow & b \\
8 & g(a(z)) & \rightarrow & b \\
10 & g(b) & \rightarrow & b
\end{array}
$$

dont on vérifie aisément qu'il est confluent.

# Deuxième stratégie équitable

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | $a(b)$ | $\rightarrow$ | $b$ | | | $\in E_0$ |
| 2 | $f(g(x))$ | $\rightarrow$ | $g(x)$ | | | $\in E_0$ |
| 3 | $g(a(x))$ | $\rightarrow$ | $b$ | | | $\in E_0$ |
| 4 | $g(x)$ | $\rightarrow$ | $f(b)$ | $\parallel$ | $x \doteq a(z)$ | $CP(2,3)$ |
| 5 | $f(b)$ | $\rightarrow$ | $b$ | | | $Simpl(3)$ |
| 6 | $g(x)$ | $\rightarrow$ | $b$ | $\parallel$ | $x \doteq a(z)$ | $Simpl(4)$ |
| 7 | $g(x)$ | $\rightarrow$ | $f(b)$ | $\parallel$ | $x \doteq a(z)$ | $CP(2,6)$ |
| 8 | $g(x)$ | $\rightarrow$ | $b$ | | $x \doteq a(z)$ | $Simpl(7)$ |
| 9 | $b$ | $=$ | $b$ | | $x \doteq a(z)$ | $Simpl(8)$ |
| 10 | | | | | | $Elim(9)$ |

La complétion est terminée, mais l'équation $g(b) = b$ n'est plus prouvable.

# Conclusion

- Cet seconde stratégie équitable de complétion simplifie la règle 3 avant le calcul de la paire critique $CP(1,3)$ qui ne sera donc pas calculée. La notion d'équité a changé : tout pic réductible doit être réduit à une étape ultérieure, pas nécessairement par le calcul de la paire critique correspondante.
- On ne peut bien sûr attendre que toutes les paires critiques d'une règle soient calculées avant de la simplifier, ce qui oterait toute vertu à la simplification.
- On va donc devoir proposer une modification de la règle de simplification, dont on devra s'assurer qu'elle fait décroître les preuves.

# Conclusion

- Cet seconde stratégie équitable de complétion simplifie la règle 3 avant le calcul de la paire critique $CP(1,3)$ qui ne sera donc pas calculée. La notion d'équité a changé : tout pic réductible doit être réduit à une étape ultérieure, pas nécessairement par le calcul de la paire critique correspondante.
- On ne peut bien sûr attendre que toutes les paires critiques d'une règle soient calculées avant de la simplifier, ce qui oterait toute vertu à la simplification.
- On va donc devoir proposer une modification de la règle de simplification, dont on devra s'assurer qu'elle fait décroître les preuves.

# Conclusion

- Cet seconde stratégie équitable de complétion simplifie la règle 3 avant le calcul de la paire critique $CP(1,3)$ qui ne sera donc pas calculée. La notion d'équité a changé : tout pic réductible doit être réduit à une étape ultérieure, pas nécessairement par le calcul de la paire critique correspondante.
- On ne peut bien sûr attendre que toutes les paires critiques d'une règle soient calculées avant de la simplifier, ce qui oterait toute vertu à la simplification.
- On va donc devoir proposer une modification de la règle de simplification, dont on devra s'assurer qu'elle fait décroître les preuves.

# Règle de simplification/instantiation des équations

$$\left\{ \begin{array}{c} l = r \parallel \phi \\ g = d \parallel \psi \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} l[d\sigma]_p = r \parallel \phi \\ g\psi \rightarrow d\psi \end{array} \right\}$$

$$\text{si } \left\{ \begin{array}{l} l \xrightarrow[g=d \parallel \psi]{p} l[d\sigma] \\ r \succ d\sigma \text{ si } l = g\sigma \end{array} \right.$$

Il est possible d'utiliser des règles de réduction plus puissantes, qui réduisent $l\phi$ au lieu de $l$, mais dont la formulation est plus complexe.

# Devoir

1. Faire la preuve de complétude du calcul obtenu en ajoutant cette règle de simplification. La règle de simplification pourra être modifiée pour accroître son pouvoir de réduction.
2. Peut-on simplifier les contraintes ? Si oui, proposer une règle et en montrer la complétude. Sinon, donner un exemple d'incomplétude.
3. Le devoir devra m'être envoyé par email en format PDF avant la publication du corrigé sur mon site le 13 janvier 2005 à 14h.

# Résolution et paramodulation ordonnées basiques

Nous allons maintenant traiter le cas de clauses arbitraires, en faisant cette fois passer les conditions d'ordre dans les contraintes, sachant que les contraintes d'ordre quantifiées existentiellement sont satisfiables.

# Règles de résolution ordonnée basique

$$\frac{+A \vee C \parallel \phi \quad -A' \vee C' \parallel \phi'}{C \vee C' \parallel \phi \wedge \phi' \wedge A \doteq A' \wedge A > C \wedge A' > C'}$$

$$\frac{+A \vee +A' \vee C \parallel \phi}{+A \vee C \parallel \phi \wedge A \doteq A' \wedge A > C}$$

$$\frac{\square \parallel \phi}{\text{Retourner } \textit{Succès} \text{ si } \phi \text{ est satisfiable}}$$

# Traitement du prédicat d'égalité

- La condition basique n'imposant aucune restriction supplémentaire en cas d'absence de traitement spécifique de l'égalité, la complétude de ces règles se déduit de la complétude des règles de résolution et factorisation ordonnées.
- On peut réduire l'espace de recherche en introduisant des mécanismes de sélection et d'élimination des redondances.
- La présence du prédicat d'égalité impose l'ajout de clauses qui forcent son interpétation comme une congruence.

# Traitement du prédicat d'égalité

- La condition basique n'imposant aucune restriction supplémentaire en cas d'absence de traitement spécifique de l'égalité, la complétude de ces règles se déduit de la complétude des règles de résolution et factorisation ordonnées.
- On peut réduire l'espace de recherche en introduisant des mécanismes de sélection et d'élimination des redondances.
- La présence du prédicat d'égalité impose l'ajout de clauses qui forcent son interpétation comme une congruence.

# Traitement du prédicat d'égalité

- La condition basique n'imposant aucune restriction supplémentaire en cas d'absence de traitement spécifique de l'égalité, la complétude de ces règles se déduit de la complétude des règles de résolution et factorisation ordonnées.
- On peut réduire l'espace de recherche en introduisant des mécanismes de sélection et d'élimination des redondances.
- La présence du prédicat d'égalité impose l'ajout de clauses qui forcent son interpétation comme une congruence.

# Axiomes de l'égalité

$$x = x \qquad \text{(réflexivité)}$$

$$x = y \ \vee \ \neg y = x \qquad \text{(symmetry)}$$

$$x = z \ \vee \ \neg x = y \vee \neg y = z \qquad \text{(transitivité)}$$

$$f(\overline{x}) = f(\overline{y}) \ \vee \ \neg \overline{x} = \overline{y} \qquad \text{(monotonie fonctionnelle)}$$

$$P(\overline{x}) = P(\overline{y}) \ \vee \ \neg \overline{x} = \overline{y} \qquad \text{(monotonie prédicative)}$$

Il y a bien sûr autant d'axiomes de monotonie fonctionelle et de monotonie prédicative que de symboles de fonction $f$ et de symboles de prédicat $P$ dans le vocabulaire.

# Exemple, avec l'ordre $rpo(P >=> a > b)$

- On se donne l'ensemble insatisfiable de clauses :

$$a = b$$
$$\neg P(b)$$
$$P(x) \quad \| \quad x \doteq a$$

- On engendre les clauses

$$x = y \quad \| \quad y = x \doteq a = b \wedge y = x > x = y$$

$$\left\{ \begin{array}{l} x = z \vee \\ \neg x = y \end{array} \right\} \quad \| \quad \left\{ \begin{array}{l} y = z \doteq a = b \wedge \\ y = z > x = z \wedge \\ y = z > x = y \end{array} \right\}$$

$$\ldots$$

- Le calcul engendre une infinité de clauses sans trouver la clause vide!

# Exemple, avec l'ordre $rpo(P >=> a > b)$

- On se donne l'ensemble insatisfiable de clauses :

$$a = b$$
$$\neg P(b)$$
$$P(x) \quad \| \quad x \doteq a$$

- On engendre les clauses

$$x = y \quad \| \quad y = x \doteq a = b \wedge y = x > x = y$$

$$\left\{ \begin{array}{l} x = z \vee \\ \neg x = y \end{array} \right\} \quad \| \quad \left\{ \begin{array}{l} y = z \doteq a = b \wedge \\ y = z > x = z \wedge \\ y = z > x = y \end{array} \right\}$$

$$\ldots$$

- Le calcul engendre une infinité de clauses sans trouver la clause vide!

# Exemple, avec l'ordre $rpo(P >=> a > b)$

- On se donne l'ensemble insatisfiable de clauses :

$$a = b$$
$$\neg P(b)$$
$$P(x) \quad \| \quad x \doteq a$$

- On engendre les clauses

$$x = y \quad \| \quad y = x \doteq a = b \wedge y = x > x = y$$

$$\left\{ \begin{array}{l} x = z \vee \\ \neg x = y \end{array} \right\} \quad \| \quad \left\{ \begin{array}{l} y = z \doteq a = b \wedge \\ y = z > x = z \wedge \\ y = z > x = y \end{array} \right\}$$

$$\ldots$$

- Le calcul engendre une infinité de clauses sans trouver la clause vide!

- L'ajout des axiomes de congruence pour chaque prédicat d'égalité conduit à une prolifération des inférences de résolution et factorisation qui engendrent un très grand nombre de clauses superflues.
- Robinson et Wos ont donc proposé de traiter l'égalité par une règle d'inférence adaptée qui élimine le besoin d'axiomatiser l'égalité :

$$\frac{C \vee l = r \qquad \pm D \vee A[u]}{C\sigma \vee D\sigma \vee \pm A[r\sigma]} \quad \text{si} \begin{cases} u \notin \mathcal{X} \\ \sigma = mgu(l = u) \end{cases}$$

- La complétude est alors préservée en présence du seul axiome de réflexivité (supposé présent dans toute la suite).

- L'ajout des axiomes de congruence pour chaque prédicat d'égalité conduit à une prolifération des inférences de résolution et factorisation qui engendrent un très grand nombre de clauses superflues.
- Robinson et Wos ont donc proposé de traiter l'égalité par une règle d'inférence adaptée qui élimine le besoin d'axiomatiser l'égalité :

$$\frac{C \vee l = r \qquad \pm D \vee A[u]}{C\sigma \vee D\sigma \vee \pm A[r\sigma]} \quad \text{si} \begin{cases} u \notin \mathcal{X} \\ \sigma = mgu(l = u) \end{cases}$$

- La complétude est alors préservée en présence du seul axiome de réflexivité (supposé présent dans toute la suite).

- L'ajout des axiomes de congruence pour chaque prédicat d'égalité conduit à une prolifération des inférences de résolution et factorisation qui engendrent un très grand nombre de clauses superflues.
- Robinson et Wos ont donc proposé de traiter l'égalité par une règle d'inférence adaptée qui élimine le besoin d'axiomatiser l'égalité :

$$\frac{C \vee l = r \qquad \pm D \vee A[u]}{C\sigma \vee D\sigma \vee \pm A[r\sigma]} \quad \text{si} \begin{cases} u \notin \mathcal{X} \\ \sigma = mgu(l = u) \end{cases}$$

- La complétude est alors préservée en présence du seul axiome de réflexivité (supposé présent dans toute la suite).

# Règles ordonnées basiques

- La paramodulation reste une règle prolifique, qui remplace des égaux par des égaux.
- Nous allons donc en étudier une version à la fois ordonnée et basique par l'emploi de contraintes adaptées.
- La règle d'élimination des clauses triviales

$$\text{Elim} \quad l = r \parallel \phi \rightarrow \mathit{nil} \quad \text{si } l\phi = r\phi$$

  restera implicite dans le système d'inférence.
- Hypothèse simplificatrice : le vocabulaire contient un unique prédicat d'égalité $=$.

# Règles ordonnées basiques

- La paramodulation reste une règle prolifique, qui remplace des égaux par des égaux.
- Nous allons donc en étudier une version à la fois ordonnée et basique par l'emploi de contraintes adaptées.
- La règle d'élimination des clauses triviales

$$\text{Elim} \quad l = r \parallel \phi \rightarrow \mathit{nil} \quad \text{si } l\phi = r\phi$$

  restera implicite dans le système d'inférence.
- Hypothèse simplificatrice : le vocabulaire contient un unique prédicat d'égalité $=$.

# Règles ordonnées basiques

- La paramodulation reste une règle prolifique, qui remplace des égaux par des égaux.
- Nous allons donc en étudier une version à la fois ordonnée et basique par l'emploi de contraintes adaptées.
- La règle d'élimination des clauses triviales

$$\text{Elim} \quad l = r \parallel \phi \rightarrow \mathit{nil} \quad \text{si } l\phi = r\phi$$

  restera implicite dans le système d'inférence.
- Hypothèse simplificatrice : le vocabulaire contient un unique prédicat d'égalité $=$.

# Règles ordonnées basiques

- La paramodulation reste une règle prolifique, qui remplace des égaux par des égaux.
- Nous allons donc en étudier une version à la fois ordonnée et basique par l'emploi de contraintes adaptées.
- La règle d'élimination des clauses triviales

$$\text{Elim} \quad l = r \parallel \phi \rightarrow \mathit{nil} \quad \text{si } l\phi = r\phi$$

  restera implicite dans le système d'inférence.
- Hypothèse simplificatrice : le vocabulaire contient un unique prédicat d'égalité $=$.

# Règles ordonnées basiques

$$\frac{+A \vee C \parallel \phi \quad -A' \vee C' \parallel \phi'}{C \vee C' \parallel \phi \wedge \phi' \wedge A \doteq A' \wedge A \succ C \wedge A' \succ C'}$$

$$\frac{+A \vee +A' \vee C \parallel \phi}{+A \vee C \parallel \phi \wedge A \doteq A' \wedge A \succ C}$$

$$\frac{C \vee l = r \parallel \phi \qquad D \vee \pm A[u] \parallel \psi}{C \vee D \vee \pm A[r] \parallel \phi \wedge \psi \wedge l = u \wedge l \succ r \wedge A \succ D}$$

où $A \succ D = \bigvee_{i \in [1..n]} \pm A_i \equiv \forall i \in [1..n]\ A \succ A_i$

$$\frac{\square \parallel \phi}{\text{Retourner } \textit{Succès} \text{ si } \phi \text{ est satisfiable}}$$

# Retour sur l'exemple avec l'ordre $rpo(a > b)$

- $$\begin{array}{rcl} P(x) & \| & x \doteq a \\ \neg P(b) & & \\ a = b & & \end{array}$$
- Cet ensemble de clauses ne contient pas la clause vide, et pourtant, aucune inférence n'est plus possible : la résolution contrainte entre les deux premières clauses engendre la contrainte insatisfiable $x \doteq a \wedge x \doteq b$, alors que la paramodulation de la seconde clause par la dernière engendre la contrainte insatisfiable $b \succ a$.
  Le système d'inférences est donc incomplet, comme la complétion basique ordonnée.

# Retour sur l'exemple avec l'ordre $rpo(a > b)$

- $$\begin{array}{rcl} P(x) & \| & x \doteq a \\ \neg P(b) & & \\ a = b & & \end{array}$$
- Cet ensemble de clauses ne contient pas la clause vide, et pourtant, aucune inférence n'est plus possible : la résolution contrainte entre les deux premières clauses engendre la contrainte insatisfiable $x \doteq a \wedge x \doteq b$, alors que la paramodulation de la seconde clause par la dernière engendre la contrainte insatisfiable $b \succ a$.
  Le système d'inférences est donc incomplet, comme la complétion basique ordonnée.

# Hypothèses

- $\succ$ est un ordre de récriture total sur les termes, les atomes et les clauses clos.
- Soit $R$ un ensemble quelconque d'atomes égalitaires clos construits sur la signature, orientés sous-forme de règles closes telles que $\longrightarrow_R^+ \subseteq \succ$. On dénote par $irred_R(\mathcal{S})$ l'ensemble des instances des clauses de $\mathcal{S}$ par une substitution close $R$-irréductible.
- Un ensemble de clauses contraintes $\mathcal{S}$ est *bien contraint* si pour tout système de règles closes $R$ confluent et tel que $\longrightarrow_R^+ \subseteq \succ$, tout modèle de $R \cup irred_R(S)$ est modèle de $S$ :

$$R \cup irred_R(\mathcal{S}) \models \mathcal{S}$$

# Hypothèses

- $\succ$ est un ordre de récriture total sur les termes, les atomes et les clauses clos.
- Soit $R$ un ensemble quelconque d'atomes égalitaires clos construits sur la signature, orientés sous-forme de règles closes telles que $\longrightarrow_R^+ \subseteq \succ$. On dénote par $irred_R(\mathcal{S})$ l'ensemble des instances des clauses de $\mathcal{S}$ par une substitution close $R$-irréductible.
- Un ensemble de clauses contraintes $\mathcal{S}$ est *bien contraint* si pour tout système de règles closes $R$ confluent et tel que $\longrightarrow_R^+ \subseteq \succ$, tout modèle de $R \cup irred_R(S)$ est modèle de $S$ :

$$R \cup irred_R(\mathcal{S}) \models \mathcal{S}$$

# Hypothèses

- $\succ$ est un ordre de récriture total sur les termes, les atomes et les clauses clos.
- Soit $R$ un ensemble quelconque d'atomes égalitaires clos construits sur la signature, orientés sous-forme de règles closes telles que $\longrightarrow_R^+ \subseteq \succ$. On dénote par $\mathit{irred}_R(\mathcal{S})$ l'ensemble des instances des clauses de $\mathcal{S}$ par une substitution close $R$-irréductible.
- Un ensemble de clauses contraintes $\mathcal{S}$ est *bien contraint* si pour tout système de règles closes $R$ confluent et tel que $\longrightarrow_R^+ \subseteq \succ$, tout modèle de $R \cup \mathit{irred}_R(S)$ est modèle de $S$ :

$$R \cup \mathit{irred}_R(\mathcal{S}) \models \mathcal{S}$$

# Exemples

- Les ensembles de clauses sans prédicat d'égalité sont-ils bien contraints ?
- Oui : car tous les systèmes $R$ sont vides.
- Les ensembles de clauses non contraintes sont-ils bien contraints ?
- Oui : étant donné $R$ qui termine, toute $\gamma$-instance d'une clause de $S$ se déduit de sa $\gamma\downarrow_R$-instance et des instances de $R$ utilisées dans la réduction.
- Les ensembles de clauses avec contraintes tautologiques sont-ils bien contraints ?
- Oui : le raisonnement est inchangé car $\gamma\downarrow_R$ satisfait toute tautologie!

# Exemples

- Les ensembles de clauses sans prédicat d'égalité sont-ils bien contraints ?
- Oui : car tous les systèmes $R$ sont vides.
- Les ensembles de clauses non contraintes sont-ils bien contraints ?
- Oui : étant donné $R$ qui termine, toute $\gamma$-instance d'une clause de $S$ se déduit de sa $\gamma{\downarrow_R}$-instance et des instances de $R$ utilisées dans la réduction.
- Les ensembles de clauses avec contraintes tautologiques sont-ils bien contraints ?
- Oui : le raisonnement est inchangé car $\gamma{\downarrow_R}$ satisfait toute tautologie!

# Exemples

- Les ensembles de clauses sans prédicat d'égalité sont-ils bien contraints ?
- Oui : car tous les systèmes $R$ sont vides.
- Les ensembles de clauses non contraintes sont-ils bien contraints ?
- Oui : étant donné $R$ qui termine, toute $\gamma$-instance d'une clause de $S$ se déduit de sa $\gamma\downarrow_R$-instance et des instances de $R$ utilisées dans la réduction.
- Les ensembles de clauses avec contraintes tautologiques sont-ils bien contraints ?
- Oui : le raisonnement est inchangé car $\gamma\downarrow_R$ satisfait toute tautologie!

# Exemples

- Les ensembles de clauses sans prédicat d'égalité sont-ils bien contraints ?
- Oui : car tous les systèmes $R$ sont vides.
- Les ensembles de clauses non contraintes sont-ils bien contraints ?
- Oui : étant donné $R$ qui termine, toute $\gamma$-instance d'une clause de $S$ se déduit de sa $\gamma{\downarrow_R}$-instance et des instances de $R$ utilisées dans la réduction.
- Les ensembles de clauses avec contraintes tautologiques sont-ils bien contraints ?
- Oui : le raisonnement est inchangé car $\gamma{\downarrow_R}$ satisfait toute tautologie!

# Exemples

- Les ensembles de clauses sans prédicat d'égalité sont-ils bien contraints ?
- Oui : car tous les systèmes $R$ sont vides.
- Les ensembles de clauses non contraintes sont-ils bien contraints ?
- Oui : étant donné $R$ qui termine, toute $\gamma$-instance d'une clause de $S$ se déduit de sa $\gamma{\downarrow_R}$-instance et des instances de $R$ utilisées dans la réduction.
- Les ensembles de clauses avec contraintes tautologiques sont-ils bien contraints ?
- Oui : le raisonnement est inchangé car $\gamma{\downarrow_R}$ satisfait toute tautologie!

# Exemples

- Les ensembles de clauses sans prédicat d'égalité sont-ils bien contraints ?
- Oui : car tous les systèmes $R$ sont vides.
- Les ensembles de clauses non contraintes sont-ils bien contraints ?
- Oui : étant donné $R$ qui termine, toute $\gamma$-instance d'une clause de $S$ se déduit de sa $\gamma{\downarrow_R}$-instance et des instances de $R$ utilisées dans la réduction.
- Les ensembles de clauses avec contraintes tautologiques sont-ils bien contraints ?
- Oui : le raisonnement est inchangé car $\gamma{\downarrow_R}$ satisfait toute tautologie!

# Exemples, suite

- Les clauses sans superposition dans leur contraintes sont-elles bien contraintes ?
- Oui : toute instance close d'une clause $C \parallel \phi$ s'écrit $C\phi\gamma$. Par hypothèse $(\phi\gamma)\downarrow_R = \phi(\gamma\downarrow_R)$ qui satisfait $\phi$. Donc $(C\phi)\gamma\downarrow_R \in \mathit{irred}_R(C)$ et le raisonnement antérieur s'applique.
- $$\left\{ \begin{array}{rcl} P(x) & \parallel & x \doteq a \\ \neg P(b) & & \\ a = b & & \end{array} \right\}$$ est-il bien contraint ?
- Non : si $R = \{a \to b\}$, alors l'instance $P(a)$ de la clause $P(x) \parallel x \doteq a$ n'est pas conséquence logique de $\{\neg P(b), a \to b\}$.

# Exemples, suite

- Les clauses sans superposition dans leur contraintes sont-elles bien contraintes ?
- Oui : toute instance close d'une clause $C \parallel \phi$ s'écrit $C\phi\gamma$. Par hypothèse $(\phi\gamma)\downarrow_R = \phi(\gamma\downarrow_R)$ qui satisfait $\phi$. Donc $(C\phi)\gamma\downarrow_R \in \mathit{irred}_R(C)$ et le raisonnement antérieur s'applique.
- $$\left\{ \begin{array}{rcl} P(x) & \parallel & x \doteq a \\ \neg P(b) & & \\ a = b & & \end{array} \right\}$$ est-il bien contraint ?
- Non : si $R = \{a \to b\}$, alors l'instance $P(a)$ de la clause $P(x) \parallel x \doteq a$ n'est pas conséquence logique de $\{\neg P(b), a \to b\}$.

# Exemples, suite

- Les clauses sans superposition dans leur contraintes sont-elles bien contraintes ?
- Oui : toute instance close d'une clause $C \parallel \phi$ s'écrit $C\phi\gamma$. Par hypothèse $(\phi\gamma)\downarrow_R = \phi(\gamma\downarrow_R)$ qui satisfait $\phi$. Donc $(C\phi)\gamma\downarrow_R \in \mathit{irred}_R(C)$ et le raisonnement antérieur s'applique.
- $$\left\{ \begin{array}{rcl} P(x) & \parallel & x \doteq a \\ \neg P(b) & & \\ a = b & & \end{array} \right\}$$ est-il bien contraint ?
- Non : si $R = \{a \to b\}$, alors l'instance $P(a)$ de la clause $P(x) \parallel x \doteq a$ n'est pas conséquence logique de $\{\neg P(b), a \to b\}$.

# Exemples, suite

- Les clauses sans superposition dans leur contraintes sont-elles bien contraintes ?
- Oui : toute instance close d'une clause $C \parallel \phi$ s'écrit $C\phi\gamma$. Par hypothèse $(\phi\gamma)\downarrow_R = \phi(\gamma\downarrow_R)$ qui satisfait $\phi$. Donc $(C\phi)\gamma\downarrow_R \in \mathit{irred}_R(C)$ et le raisonnement antérieur s'applique.
- $$\left\{ \begin{array}{rcl} P(x) & \parallel & x \doteq a \\ \neg P(b) & & \\ a = b & & \end{array} \right\}$$ est-il bien contraint ?
- Non : si $R = \{a \to b\}$, alors l'instance $P(a)$ de la clause $P(x) \parallel x \doteq a$ n'est pas conséquence logique de $\{\neg P(b), a \to b\}$.

# Exemples, fin

- $\mathcal{S} = \{g(x,x) = b \parallel a > x\}$ est-il bien contraint ?
- Oui. On distingue deux cas.
  Si $b \succ a$, alors $a$ est minimal et $\mathcal{S} = \emptyset$ qui est bien contraint.
  Si $a \succ b$, alors, pour toute instance close $g(u,u) = b$ telle que $a \succ u$ et $u \longrightarrow_R^* v = u{\downarrow_R}$, alors $a \succ u \succeq v$ et donc $a \succ v$ par transitivité. Donc $v$ satisfait la contrainte, et $g(v,v) = b \in \mathit{irred}_R(S)$. Or, $\{g(v,v) = b\} \cup R \models g(u,u)$.
  Notons que si $\succ = \mathit{rpo}(g > a > f > b)$, alors $\mathcal{S}$ dénote l'ensemble non reconnaissable de clauses $\{g(f^n(b), f^n(b)) = b \mid n \in \mathbb{N}\}$.

# Exemples, fin

- $\mathcal{S} = \{g(x,x) = b \parallel a > x\}$ est-il bien contraint ?
- Oui. On distingue deux cas.
  Si $b \succ a$, alors $a$ est minimal et $\mathcal{S} = \emptyset$ qui est bien contraint.
  Si $a \succ b$, alors, pour toute instance close $g(u,u) = b$ telle que $a \succ u$ et $u \longrightarrow_R^* v = u\downarrow_R$, alors $a \succ u \succeq v$ et donc $a \succ v$ par transitivité. Donc $v$ satisfait la contrainte, et $g(v,v) = b \in \mathit{irred}_R(S)$. Or, $\{g(v,v) = b\} \cup R \models g(u,u)$.
  Notons que si $\succ = \mathit{rpo}(g > a > f > b)$, alors $\mathcal{S}$ dénote l'ensemble non reconnaissable de clauses $\{g(f^n(b), f^n(b)) = b \mid n \in \mathbb{N}\}$.

# Complétude

## Definition

Un ensemble $\mathcal{S}$ de clauses est dit *clos* pour un ensemble de règles d'inférences $\mathcal{I}$ si toute clause contrainte $C \parallel \phi$ inférable par $\mathcal{I}$ à partir de clauses de $\mathcal{S}$ appartient à $\mathcal{S}$.

## Theorem

*Tout ensemble de clauses $\mathcal{S}$ bien contraint et clos est insatisfiable ssi il contient la clause vide.*

On suppose pour simplifier que $=$ est l'unique symbole de prédicat

# Le système de récriture clos $R_{\mathcal{S}}$

Un ensemble de règles $R$ définit un modèle unique sur l'univers de Herbrand en interprétant $=$ par $\longleftrightarrow_R^*$, qui est isomorphe à $T(F)/=_R$.

## Definition

La clause $C = C' \vee l = r$ instance d'une clause contrainte de $\mathcal{S}$ par la substitution close $\gamma$ *produit* la règle $l \to r$ ssi :

1. $l \succ r$ et $l \succ u$ pour tout terme $u$ de $C'$
2. $l$ et $\gamma$ sont irréductibles par $R_C = \{g \to d \mid \exists D \prec C$ qui produit $g \to d\}$
3. $R_C \not\models C$ (aucun modèle de $R_C$ ne valide $C$)

$R_{\mathcal{S}}$ est l'ensemble des règles produites par $\mathcal{S}$.

# Exemple

On se donne le vocabulaire $\mathcal{F} = \{a, b, f()\}$ et l'ensemble de clauses:

$$f(a) = y \vee f(y) = y$$

On choisit pour ordre sur les termes le rpo engendré par la précédence $f > a > b$, permettant d'énumérer les terms dans l'ordre croissant comme suit:

$$b \;\; a \;\; f(b) \;\; f(a) \;\; f(f(b)) \;\; f(f(a)) \;\; f(f(f(b))) \ldots$$

Il y a alors deux règles produites:

$$\{f(a) \to b, \quad f(f(b)) \to f(b)\}$$

# Propriété fondamentale de $R_S$

**Lemma**

*$R_S$ est un système clos confluent qui termine.*

On notera $C\downarrow$ pour $C\downarrow_{R_S}$.

**Proof.**

Soient $C, D$ deux clauses telles que $C \succ D$, $l \to r \in R_C$ et $g \to d \in R_D$.
Par définition de $R_C$, $g$ ne peut être réductible par $l \to r$ et comme $l$ et $g$ sont clos, $g$ ne peut-être sous-terme (au sens large) de $l$.
Il faut donc que la définition de l'ordre $\succ$ sur les clauses assure que $l$ ne peut être sous-terme strict de $g$. □

# Preuve de complétude

**Lemma**

*Soit $\mathcal{S}$ un ensemble de clauses bien contraint clos ne contenant pas la clause vide. Alors $R_{\mathcal{S}} \models \mathit{irred}_{R_{\mathcal{S}}}(\mathcal{S})$.*

La preuve de ce lemme vient après celle du théorème.

**Preuve du théorème :**

Si $\square \notin \mathcal{S}$, alors $R_{\mathcal{S}} \models \mathit{irred}_{R_{\mathcal{S}}}(\mathcal{S}) \models \mathcal{S}$ puisque $\mathcal{S}$ est bien contraint pour $R_{\mathcal{S}}$, et $\mathcal{S}$ est donc satisfiable. $\square$

# Preuve du lemme

On raisonne par contradiction, ce qui implique l'existence d'une clause $C\gamma \in \mathit{irred}_{R_\mathcal{S}}(\mathcal{S})$, que nous choisissons minimale pour $\succ$, qui soit l'instance d'une clause $C \parallel \phi$ par la substitution close $\gamma = \gamma{\downarrow_{R_\mathcal{S}}}$ et telle que $R_\mathcal{S} \not\models C\gamma$.
On va montrer qu'une inférence est possible qui construit une nouvelle clause $C' \parallel \phi'$ telle que $\gamma$ satisfait $\phi'$ et donc $C'\gamma \in \mathit{irred}_{R_\mathcal{S}}(\mathcal{S})$, $R_\mathcal{S} \not\models C'\gamma$, et $C\gamma \succ C'\gamma$, contredisant l'existence de $C\gamma$.
On considère les différents cas suivant la forme du terme maximal $s\gamma$ de $C\gamma$.

# Preuve du lemme, cas 1 et 2

1. $C = B \vee s = t \parallel \phi$ et $s\gamma = t\gamma$. Ce cas n'est pas possible puisque $R_{\mathcal{S}} \not\models C\gamma$.
2. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma = t\gamma$. Par résolution contrainte de $C$ avec la clause $x = x$, on engendre la clause contrainte $B \parallel \phi \wedge s \doteq t$ qui appartient donc à $\mathcal{S}$ par hypothèse, dont l'instance close $B\gamma$ vérifie les propriétés annoncées : $B\gamma \in \mathit{irred}_{R_{\mathcal{S}}}(\mathcal{S})$ ($\gamma$ est irréductible) , $C\gamma \succ B\gamma$ (propriété de l'ordre) et $R_{\mathcal{S}} \not\models B\gamma$ (puisque $R_{\mathcal{S}} \not\models B\gamma \vee \neg s\gamma = t\gamma$). Ce cas est donc lui-aussi impossible.

# Preuve du lemme, cas 1 et 2

1. $C = B \vee s = t \parallel \phi$ et $s\gamma = t\gamma$. Ce cas n'est pas possible puisque $R_{\mathcal{S}} \not\models C\gamma$.
2. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma = t\gamma$. Par résolution contrainte de $C$ avec la clause $x = x$, on engendre la clause contrainte $B \parallel \phi \wedge s \doteq t$ qui appartient donc à $\mathcal{S}$ par hypothèse, dont l'instance close $B\gamma$ vérifie les propriétés annoncées : $B\gamma \in \mathit{irred}_{R_{\mathcal{S}}}(\mathcal{S})$ ($\gamma$ est irréductible) , $C\gamma \succ B\gamma$ (propriété de l'ordre) et $R_{\mathcal{S}} \not\models B\gamma$ (puisque $R_{\mathcal{S}} \not\models B\gamma \vee \neg s\gamma = t\gamma$). Ce cas est donc lui-aussi impossible.

# Preuve du lemme, cas 3

3. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma \succ t\gamma$.
   Comme $R_{\mathcal{S}} \not\models C\gamma$, $R_{\mathcal{S}} \models s\gamma = t\gamma$. Il y a donc une preuve par récriture de $s\gamma = t\gamma$ avec $R_{\mathcal{S}}$, et comme $s\gamma \succ t\gamma$, $s\gamma$ est réductible à la position $p$ par une règle $l\gamma \rightarrow r\gamma \in R_{\mathcal{S}}$ produite par la clause $D \vee l = r \parallel \psi$. Comme $\gamma$ est irréductible (pour les variables de $C$) $p \in \mathcal{FP}os(s)$, et donc $(s|_p)\gamma = l\gamma$ puisque $l\gamma$ est clos. On peut donc faire l'inférence :

$$\frac{B \vee \neg s = t \parallel \phi \qquad D \vee l = r \parallel \psi}{B \vee D \vee \neg s[r]_p = t \parallel \phi \wedge \psi \wedge s|_p \doteq l \wedge s = t > B}$$

# Preuve du lemme, suite et fin

La clause obtenue appartient à $\mathcal{S}$ puisque $\mathcal{S}$ est clos par inférence. Par maximalité de $s$, la substitution en forme normale $\gamma$ satisfait sa contrainte. Son instance $B\gamma \vee D\gamma \vee s\gamma[r\gamma]_p = t\gamma$ appartient donc à $\mathit{irred}_{R_\mathcal{S}}(\mathcal{S})$. Elle est enfin plus petite que la clause $B\gamma \vee \neg s\gamma = t\gamma$ par maximalité de $s$ dans $C\gamma$ (par hypothèse) et de $l\gamma$ dans $D\gamma$ (par définition de $R_\mathcal{S}$).

- $C = B \vee s = t \parallel \phi$ avec $s\gamma \succ t\gamma$. Comme $R_\mathcal{S} \not\vdash C\gamma$, $R_\mathcal{S} \not\vdash s\gamma = t\gamma$. Donc, il doit exister une instance de clause plus petite $D\gamma \vee l\gamma = r\gamma$ qui a engendré $g\gamma = d\gamma$ telle que $s\gamma \to t\gamma$ soit réductible par $l\gamma \to r\gamma$. On conclue alors comme dans le cas précédent

# Preuve du lemme, suite et fin

La clause obtenue appartient à $\mathcal{S}$ puisque $\mathcal{S}$ est clos par inférence. Par maximalité de $s$, la substitution en forme normale $\gamma$ satisfait sa contrainte. Son instance $B\gamma \vee D\gamma \vee s\gamma[r\gamma]_p = t\gamma$ appartient donc à $irred_{R_\mathcal{S}}(\mathcal{S})$. Elle est enfin plus petite que la clause $B\gamma \vee \neg s\gamma = t\gamma$ par maximalité de $s$ dans $C\gamma$ (par hypothèse) et de $l\gamma$ dans $D\gamma$ (par définition de $R_\mathcal{S}$).

4. $C = B \vee s = t \parallel \phi$ avec $s\gamma \succ t\gamma$. Comme $R_\mathcal{S} \not\models C\gamma$, $R_\mathcal{S} \not\models s\gamma = t\gamma$. Donc, il doit exister une instance de clause plus petite $D\gamma \vee l\gamma = r\gamma$ qui a engendré $g\gamma = d\gamma$ telle que $s\gamma \rightarrow t\gamma$ soit réductible par $l\gamma \rightarrow r\gamma$. On conclue alors comme dans le cas précédent.

# Cas général

On a supposé que $=$ est l'unique symbole de prédicat. On peut traiter le cas général de deux manières différentes:
On transforme les autres symboles de prédicats en fonctions à valeur dans Bool et on montre que le codage de l'atome $P(\bar{t})$ par l'égalité $P(\bar{t}) = T$ préserve la satisfiabilité.
On traite directement le cas général.

## Definition

La clause $C = C' \vee l = r$ (resp. $C' \vee A$) instance d'une clause contrainte de $\mathcal{S}$ par la substitution close $\gamma$ *produit* la règle $l \to r$ (resp. l'atome $A$) ssi :

1. $l = r \succ B$ (resp. $A \succ B$) pour tout atome $B$ de $C'$
2. $l$ (resp. $A$) et $\gamma$ sont irréductibles par $R_C = \{g \to d \mid \exists D \prec C$ qui produit $g \to d\}$; $A_C$ est l'ensemble des atomes produits ;
3. Aucun modèle de $R_C \cup A_C$ ne valide $C$

$R_\mathcal{S}$ et $A_\mathcal{S}$ sont les ensembles de règles et atomes produits par $\mathcal{S}$.

# Exemple

On se donne le vocabulaire $\mathcal{F} = \{a, b, f()\}$ et $\mathcal{P} = \{P()\}$, et l' ensemble de clauses:

$$P(f(x))$$
$$\neg P(b)$$
$$f(a) = y \vee f(y) = y$$

On choisit pour ordre sur les atomes le rpo engendré par $P > f > a > b$, qui énumére les atomes non triviaux comme suit:

$$b = a \;\; b = f(b) \;\; a = f(b) \;\; b = f(a) \;\; a = f(a) \;\; b = f(f$$
$$P(b) \;\; P(a) \;\; P(f(b)) \;\; P(f(a)) \;\; P(f(f(b))) \;\; P(f(f(a))) \, .$$

Il y a alors deux règles et trois atomes produits:

$$\{f(a) \to b \quad f(f(b)) \to f(b)\} \quad \{P(b) \;\; P(a) \;\; P(f(b))\}$$

# Preuve de complétude

**Lemma**

*Soit $\mathcal{S}$ un ensemble de clauses bien contraint clos ne contenant pas la clause vide. Alors $R_{\mathcal{S}} \cup A_{\mathcal{S}} \models irred_{R_{\mathcal{S}}}(\mathcal{S})$.*

La preuve de ce lemme vient après celle du théorème.

**Preuve du théorème :**

Si $\square \notin \mathcal{S}$, alors $R_{\mathcal{S}} \cup A_{\mathcal{S}} \models irred_{R_{\mathcal{S}}}(\mathcal{S}) \models \mathcal{S}$ puisque $\mathcal{S}$ est bien contraint pour $R_{\mathcal{S}}$, et $\mathcal{S}$ est donc satisfiable. $\square$

# Preuve du lemme

On raisonne par contradiction, ce qui implique l'existence d'une clause $C\gamma \in \mathit{irred}_{R_\mathcal{S}}(\mathcal{S})$, que nous choisissons minimale pour $\succ$, qui soit l'instance d'une clause $C \parallel \phi$ par la substitution close $\gamma = \gamma{\downarrow_{R_\mathcal{S}}}$ et telle que $R_\mathcal{S} \cup A_\mathcal{S} \not\models C\gamma$.

On va montrer qu'une inférence est possible qui construit une nouvelle clause $C' \parallel \phi'$ telle que $\gamma$ satisfait $\phi'$ et donc $C'\gamma \in \mathit{irred}_{R_\mathcal{S}}(\mathcal{S})$, $R_\mathcal{S} \cup A_\mathcal{S} \not\models C'\gamma$, et $C\gamma \succ C'\gamma$, contredisant l'existence de $C\gamma$.
On considère les différents cas suivant la forme de l'atome maximal $s\gamma$ de $C\gamma$.

# Cas inchangés

1. $C = B \vee s = t \parallel \phi$ et $s\gamma = t\gamma$.
2. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma = t\gamma$.
3. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma \succ t\gamma$.
4. $C = B \vee s = t \parallel \phi$ avec $s\gamma \succ t\gamma$.

# Cas inchangés

1. $C = B \vee s = t \parallel \phi$ et $s\gamma = t\gamma$.
2. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma = t\gamma$.
3. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma \succ t\gamma$.
4. $C = B \vee s = t \parallel \phi$ avec $s\gamma \succ t\gamma$.

# Cas inchangés

1. $C = B \vee s = t \parallel \phi$ et $s\gamma = t\gamma$.
2. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma = t\gamma$.
3. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma \succ t\gamma$.
4. $C = B \vee s = t \parallel \phi$ avec $s\gamma \succ t\gamma$.

# Cas inchangés

1. $C = B \vee s = t \parallel \phi$ et $s\gamma = t\gamma$.
2. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma = t\gamma$.
3. $C = B \vee \neg s = t \parallel \phi$ et $s\gamma \succ t\gamma$.
4. $C = B \vee s = t \parallel \phi$ avec $s\gamma \succ t\gamma$.

# Nouveau cas $C = B \vee \neg A \parallel \phi$

Comme $R_{\mathcal{S}} \cup A_{\mathcal{S}} \not\models C\gamma$, $A$ s'interprète en vrai dans tout modèle de $R_{\mathcal{S}} \cup A_{\mathcal{S}}$. On distingue deux cas:

1. Si $A$ est réductible par $R_{\mathcal{S}}$, alors on conclue par clôture par paramodulation contrainte comme précédemment.
2. Sinon, $A \in A_{\mathcal{S}}$, et donc une clause de la forme $D \vee B \parallel \psi$ a produit $A$ via la substitution $\gamma$ en forme normale pour $R_{\mathcal{S}}$. On conclue par résolution contrainte avec cette clause.

# Nouveau cas $C = B \vee A \parallel \phi$

Comme $R_{\mathcal{S}} \cup A_{\mathcal{S}} \not\models C\gamma$, $A$ s'interprète en faux dans tout modèle de $R\mathcal{S} \cup A_{\mathcal{S}}$, et donc $A \notin \mathcal{A}_{\mathcal{S}}$. Par d'efinition de $A_{\mathcal{S}}$, on en déduit l'existence d'une règle produite par la clause $D \vee l = r \parallel \psi$ qui réduit $A$ avec laquelle on va faire une paramodulation contrainte comme précédemment.

# Élimination des redondances

# Exemples de redondances

On va maintenant rajouter des règles de

- simplification
- subsomption
- élimination des clauses triviales
- élimination des tautologies
- etc.

La notion de redondance va nous permettre de les traiter de manière uniforme, en définissant une notion de clôture “à redondance près”, appellée *saturation*.

# Redondances en avant et en arrière

L'application d'une règle de redondance est dite

- *en avant* si elle permet d'éliminer une clause engendrée à l'aide de clauses préexistantes.
- *en arrière* si elle permet d'éliminer une clause existante avec des clauses engendrées ultérieurement.

Subsomption et simplification peuvent être suivant les cas en avant ou en arrière.
L'élimination des tautologies et des clauses triviales est en avant.
L'application de règles de redondance en arrière est souvent plus délicate du point de vue de la complétude.

# Redondance : exemple

On choisit l'ordre $lpo(f \succ a \succ b)$.

1. $f(a,x) = x \quad \in S_0$
2. $f(x,a) = f(x,b) \quad \in S_0$

3. $f(x,b) = y \parallel x \doteq a \wedge y \doteq a$ Sup(1,2)
4. $z = y \parallel \begin{cases} x \doteq a \wedge y \doteq a \\ \wedge z \doteq b \end{cases}$ Sup(1,3)
5. $y = z \parallel y \doteq a \wedge z \doteq b$ Clean(4)
6. $f(b,x) = x$ Simpl(1,5)
7. $f(x,b) = f(x,b)$ Simpl(2,5)
8. Élimination de la tautologie 7

L'ensemble $\{3,5,6\}$ de règles contraintes obtenu est confluent.

# Redondance : exemple

On choisit l'ordre $lpo(f \succ a \succ b)$.

1. $f(a,x) = x \quad \in S_0$
2. $f(x,a) = f(x,b) \quad \in S_0$

| | | | | |
|---|---|---|---|---|
| 3. | $f(x,b) = y$ | $\parallel$ | $x \doteq a \wedge y \doteq a$ | Sup(1,2) |
| 4. | $z = y$ | $\parallel$ | $\begin{cases} x \doteq a \wedge y \doteq a \\ \wedge z \doteq b \end{cases}$ | Sup(1,3) |
| 5. | $y = z$ | $\parallel$ | $y \doteq a \wedge z \doteq b$ | Clean(4) |
| 6. | $f(b,x) = x$ | | | Simpl(1,5) |
| 7. | $f(x,b) = f(x,b)$ | | | Simpl(2,5) |
| 8. | Élimination de la tautologie 7 | | | |

L'ensemble $\{3,5,6\}$ de règles contraintes obtenu est confluent.

# Redondance et saturation

## Definition

Une clause $C \parallel \phi \in \mathcal{S}$ est *redondante* si $C\gamma$ est *redondante* dans $\mathcal{S}$ pour toute substitution close $\gamma$ satisfaisant $\phi$, c'est-à-dire est instance close d'une clause de $\mathcal{S} \setminus \{C \parallel \phi\}$ ou conséquence logique d'instances closes strictement plus petites de clauses de $\mathcal{S} \setminus \{C \parallel \phi\}$.
Un ensemble $\mathcal{S}$ est *saturé* si toute clause inférable à partir de $\mathcal{S}$ est redondante dans $\mathcal{S}$.

## Theorem

*Tout ensemble de clauses $\mathcal{S}$ bien contraint saturé est insatisfiable ssi il contient la clause $\square$.*

# Complétude

**Proof.**

On reprend la preuve de complétude dans le cas où $\mathcal{S}$ ne contient pas $\square$. Le premier cas est inchangé. Pour les autres, la $\gamma$-instance close irréductible de la clause engendrée est soit une instance close d'une clause de $\mathcal{S}$, et la minimalité de $C\gamma$ est contredite, ou bien est conséquence logique d'instances closes de clauses de $\mathcal{S}$ plus petites. Comme $\mathcal{S}$ est bien contraint, on peut les supposer par minimalité dans $\mathit{irred}_{R_\mathcal{S}}(\mathcal{S})$. Comme $R_\mathcal{S}$ n'est pas modèle de la clause engendrée, celles dont il n'est pas modèle contredisent la minimalité de $C\gamma$. $\square$

# Exemple

On choisit l'ordre $lpo(f > h > g > a)$.

$$
\begin{array}{lrcll}
1. & g(x) & = & x & (\in S_0) \\
2. & h(a,z) & = & z & (\in S_0) \\
3. & f(x,h(x,y)) & = & g(y) & (\in S_0) \\
4. & f(a,z) & = & z & (\in S_0)
\end{array}
$$

$$
5.\quad f(x,z) = g(y) \quad \| \quad \begin{array}{l} x \doteq a \wedge y \doteq z \wedge \\ h(a,z) > z \wedge \\ f(x,h(x,z)) > g(z) \end{array} \qquad (2,3)
$$

Cette inférence est en fait redondante, car ses instances closes $f(a,t) = g(t)$ se déduisent par récriture des instances closes plus petites $f(a,t) = t$ de 4, et $g(t) = t$ de 1.

# Exemple

On choisit l'ordre $lpo(f > h > g > a)$.

$$
\begin{array}{lrcll}
1. & g(x) & = & x & (\in S_0) \\
2. & h(a,z) & = & z & (\in S_0) \\
3. & f(x,h(x,y)) & = & g(y) & (\in S_0) \\
4. & f(a,z) & = & z & (\in S_0)
\end{array}
$$

$$
5.\quad f(x,z) = g(y) \quad \| \quad \begin{array}{l} x \doteq a \wedge y \doteq z \wedge \\ h(a,z) > z \wedge \\ f(x,h(x,z)) > g(z) \end{array} \qquad (2,3)
$$

Cette inférence est en fait redondante, car ses instances closes $f(a,t) = g(t)$ se déduisent par récriture des instances closes plus petites $f(a,t) = t$ de 4, et $g(t) = t$ de 1.

# Autre exemple

Soit $S_0 = p(x,z) \vee \neg p(x,y) \vee \neg p(y,z)\}$, avec l'ordre $lpo(b \succ a \succ c \succ d)$. Par résolution ordonnée contrainte, on engendre :

$$\frac{\left\{\begin{array}{c} p(x,u) \vee \neg p(x,y) \vee \neg p(y,u) \\ p(y,u) \vee \neg p(y,z) \vee \neg p(z,u) \end{array}\right\}}{\left\{\begin{array}{c} p(x,u) \vee \neg p(x,y) \vee \neg p(y,z) \vee \neg p(z,u) \\ \| y > x \wedge u > z \wedge y > z \end{array}\right\}}$$

dont les instances closes sont conséquences des instances closes plus petites :

1. $p(b,d) \vee \neg p(b,c) \vee \neg p(c,d)$
2. $p(a,d) \vee \neg p(a,b) \vee \neg p(b,d)$
3. $p(a,c) \vee \neg p(a,b) \vee \neg p(b,c)$
4. $p(a,d) \vee \neg p(a,c) \vee \neg p(c,d)$

# Autre exemple

Soit $S_0 = p(x,z) \vee \neg p(x,y) \vee \neg p(y,z)\}$, avec l'ordre $lpo(b \succ a \succ c \succ d)$. Par résolution ordonnée contrainte, on engendre :

$$\frac{\left\{\begin{array}{c} p(x,u) \vee \neg p(x,y) \vee \neg p(y,u) \\ p(y,u) \vee \neg p(y,z) \vee \neg p(z,u) \end{array}\right\}}{\left\{\begin{array}{c} p(x,u) \vee \neg p(x,y) \vee \neg p(y,z) \vee \neg p(z,u) \\ \| y > x \wedge u > z \wedge y > z \end{array}\right\}}$$

dont les instances closes sont conséquences des instances closes plus petites :

1. $p(b,d) \vee \neg p(b,c) \vee \neg p(c,d)$
2. $p(a,d) \vee \neg p(a,b) \vee \neg p(b,d)$
3. $p(a,c) \vee \neg p(a,b) \vee \neg p(b,c)$
4. $p(a,d) \vee \neg p(a,c) \vee \neg p(c,d)$

# Calcul d'ensembles saturés

## Definition

Une *derivation* est une suite $\{S_i\}_i$ d'ensemble de clauses contraintes t.q. $S_{i+1} = S_i \cup \{C \parallel \phi\}$ où $C \parallel \phi$ est inférée à partir de clauses de $S_i$, ou $S_{i+1} = S_i \setminus \{C \parallel \phi\}$ où $C \parallel \phi$ est redondante.

On note $S^* = \bigcup_i S_i$ et $S^\infty = \bigcup_i \bigcap_{j \geq i} S_i$

## Lemma

*Soit $\{S_i\}_i$ une dérivation, et $C \parallel \phi \in S^* \setminus S^\infty$. Alors $C$ est redondante dans $S^\infty$.*

## Proof.

Par récurrence sur l'ordre sur les clauses. □

# Équité du calcul d'ensembles saturés

## Definition

Une dérivation $\{S_i\}_i$ est *équitable* si pour toute inférence dont les prémisses sont dans $S^\infty$, la conclusion est dans $S^*$.

## Theorem

*Pour toute dérivation équitable, $S^\infty$ est un ensemble de clauses saturé logiquement équivalent à $S_0$.*

## Proof.

C'est une conséquence du lemme précédent et de la notion d'équité. □

# Redondance et saturation : nouvel exemple

| | | |
|---|---|---|
| 1. | $a = b$ | $(\in S_0)$ |
| 2. | $f(g(x)) = g(x)$ | $(\in S_0)$ |
| 3. | $f(g(a)) = b$ | $(\in S_0)$ |
| 4. | $\neg g(b) = b$ | $(\in S_0)$ |
| 5. | $g(x) = b \parallel x \doteq a$ | (Superpose 3 sur 2) |
| 6. | $f(b) = b$ | (Simplifie 3 par 5) |
| 7. | $f(b) = g(x) \parallel x \doteq a$ | (Superpose 5 sur 2) |
| 8. | $g(x) = b \parallel x \doteq a$ | (Simplifie 7 par 6) |
| 9. | $b = b \parallel x \doteq a$ | (Simplifie 8 par 5) |
| 10. | Suppression de 9 | |

L'ensemble de clauses $\{1, 2, 4, 5, 6\}$ est insatisfiable et saturé mais ne contient pas $\square$.

# Redondance et saturation : nouvel exemple

| | | | |
|---|---|---|---|
| 1. | $a = b$ | | $(\in S_0)$ |
| 2. | $f(g(x)) = g(x)$ | | $(\in S_0)$ |
| 3. | $f(g(a)) = b$ | | $(\in S_0)$ |
| 4. | $\neg g(b) = b$ | | $(\in S_0)$ |
| 5. | $g(x) = b$ | $\parallel \; x \doteq a$ | (Superpose 3 sur 2) |
| 6. | $f(b) = b$ | | (Simplifie 3 par 5) |
| 7. | $f(b) = g(x)$ | $\parallel \; x \doteq a$ | (Superpose 5 sur 2) |
| 8. | $g(x) = b$ | $\parallel \; x \doteq a$ | (Simplifie 7 par 6) |
| 9. | $b = b$ | $\parallel \; x \doteq a$ | (Simplifie 8 par 5) |
| 10. | Suppression de 9 | | |

L'ensemble de clauses $\{1, 2, 4, 5, 6\}$ est insatisfiable et saturé mais ne contient pas $\square$.

# Simplifications admissibles

Où est le problème ?

# Simplifications admissibles

On se propose de passer en revue les règles qui conservent la propriété d'ensemble bien contraint.

1. Subsomption : oui, car on ne fait qu'éliminer.
2. Simplification dans les règles : non.
   Notons que la simplification est traitée en 2 temps : ajout de la règle simplifiée, puis retrait de la première.
3. Simplification dans les contraintes : non.
4. Instantiation des règles : oui.

Il peut être judicieux d'instancier avant de simplifier si la règle engendrée n'est pas bien contrainte.

# Exemple revu

Il faut s'assurer que les règles de simplification suppriment des clauses redondantes, mais que l'ensemble de clauses saturé obtenu est bien contraint. Le problème est que la propriété n'est pas monotone. Simplifions les contraintes :

| | | | | |
|---|---|---|---|---|
| 1. | $a = b$ | | | $(\in S_0)$ |
| 2. | $f(g(x)) = g(x)$ | | | $(\in S_0)$ |
| 3. | $f(g(a)) = b$ | | | $(\in S_0)$ |
| 4. | $\neg g(b) = b$ | | | $(\in S_0)$ |
| 5. | $g(x) = b$ | $\|\|$ | $x \doteq a$ | (Superpose 3 sur 2) |
| 6. | $f(b) = b$ | | | (Simplification de 3) |
| 7. | $g(x) = b$ | $\|\|$ | $x \doteq b$ | (Simplification de 5) |
| 8. | $\square$ | $\|\|$ | $x \doteq b$ | (Résolution de 4 et 7) |

# Règles de simplification

La simplification dans les contraintes ne résoud pas le problème en général. Un complément utile est l'instantiation des clauses "mal contraintes" de manière à provoquer l'apparition de nouvelles inférences. On voit que le problème est complexe, et reste un compromis entre des exigences antagonistes.

**Theorem**

*Pour toute dérivation équitable pour laquelle $S^\infty$ est un ensemble de clauses bien contraintes, $S_0$ est insatisfiable ssi $S^\infty$ contient la clause vide.*