Réécriture

Complétion

Le lemme de Knuth-Bendix et la confluence locale

1 Paires critiques

Si il existe

- deux règles $l \to r$ et $g \to d$,
- une position $p \in Pos(l)$,
- un mgu σ de $l_{|p}$ et de g.

Le terme $\sigma(l)$ est applé une superposition
et la paire $\langle \sigma(r) = \sigma(\sigma(l[d]_p)) \rangle$ est appelée une paire critique.

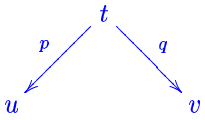
Problème :
Le problème est de savoir si les paires critiques sont joignables .

2 Théorème de Knuth et Bendix

Etant donné un système de réécriture R. R est confluent si et seulement si toutes ses pairs critiques sont confluentes.

Démonstration

- La condition nécessaire est clairement suffisante.
- Pour la condition suffisante, il faut examiner les triplets



et considérer les positions respectives de p et q et les régles $l \to r$ et $g \to d$ qui permettent de réécrire t.

- Si p et q sont étrangers alors (voir dessin)

- $u \xrightarrow[l \to r, \theta]{p} u'$,
- $u \xrightarrow[g \to d, \rho]{q} v'$
- et $u' \equiv v'$.

– Si p est un préfixe de q (avec $q \equiv pq'$) et $q \in Pos(l)$, on a le résultat parce que la paire critique est convergente (voir dessin).

– Si p est un préfixe de q (avec $q \equiv pq'$) et $q \notin Pos(l)$, on peut réécrire les "résidus" (voir dessin).

Un exemple

3 associativité + endomorphisme

$$(x * y) * z \rightarrow x * (y * z) \quad (1)$$
$$f(x) * f(y) \rightarrow f(x * y) \quad (2)$$

Exemples de paires critiques :

$$(f(x) * f(y)) * z \xrightarrow{1} f(x) * (f(y) * z)$$
$$\xrightarrow{2} f(x * y) * z$$

$(f(x) * f(y)) * z$ est une superposition

$$\langle f(x) * (f(y) * z), f(x * y) * z \rangle$$

est une paire critique **divergente** (ou non joignable).

$$\langle (x_1 * (x_2 * y)) * z, (x_1 * x_2) * (y * z) \rangle$$

est une paire critique **convergente** (ou joignable) , puisque les deux membres se réécrivent vers le même terme.

4 Procédure de Complétion

Une procédure de complétion transforme un ensemble de d'identités en un ensemble équivalent de règles confluentes et qui se terminent.

Plus généralement, une procédure de complétion transforme un ensemble de d'identités et règles confluentes et qui se terminent en un ensemble équivalent de règles confluentes et qui se terminent.

Cela nécessite un ordre pour orienter les identités.

4.1 Notion d'enchâssement

La relation d'enchâssement $\triangleright$ (en anglais encompassment) est définie par

$$s \triangleright t$$

si et seulement si

$$s \neq t \quad \text{et} \quad \exists p \in Pos(s) \cdot \exists \sigma \in Sub(T(\Sigma, V)) \cdot \quad s_{|p} = \sigma(t)$$

Résultat: $\triangleright$ termine.

Remarque Voir le tp pour savoir exactement quelle relation (en fait celle du All that) termine .

4.2 Procédure de complétion

Delete: $E \cup \{s = s\}; R \quad \vdash \quad E; R$

Compose: $E; R \cup \{s \to t\} \quad \vdash \quad E; R \cup \{s \to u\}$
si $t \xrightarrow[R]{} u$

Simplify: $E \cup \{s = t\}; R \quad \vdash \quad E \cup \{s = u\}; R$
si $t \xrightarrow[R]{} u$

Collapse: $E; R \cup \{s \to t\} \quad \vdash \quad E \cup \{u = t\}; R$
si $s \xrightarrow[R]{} u$
par une règle $l \to r \in R$
avec $s \triangleright l$
ou $s = l$ et $t > r$

Orient: $E \cup \{s = t\}; R \quad \vdash \quad E; R \cup \{s \to t\}$
si $s > t$

Deduce: $E; R \quad \vdash \quad E \cup \{s = t\}; R$
si $s \leftarrow_R u \xrightarrow[R]{} t$
pour un u

5 Complétion de l'*associativité + endomorphisme*

Commençons par

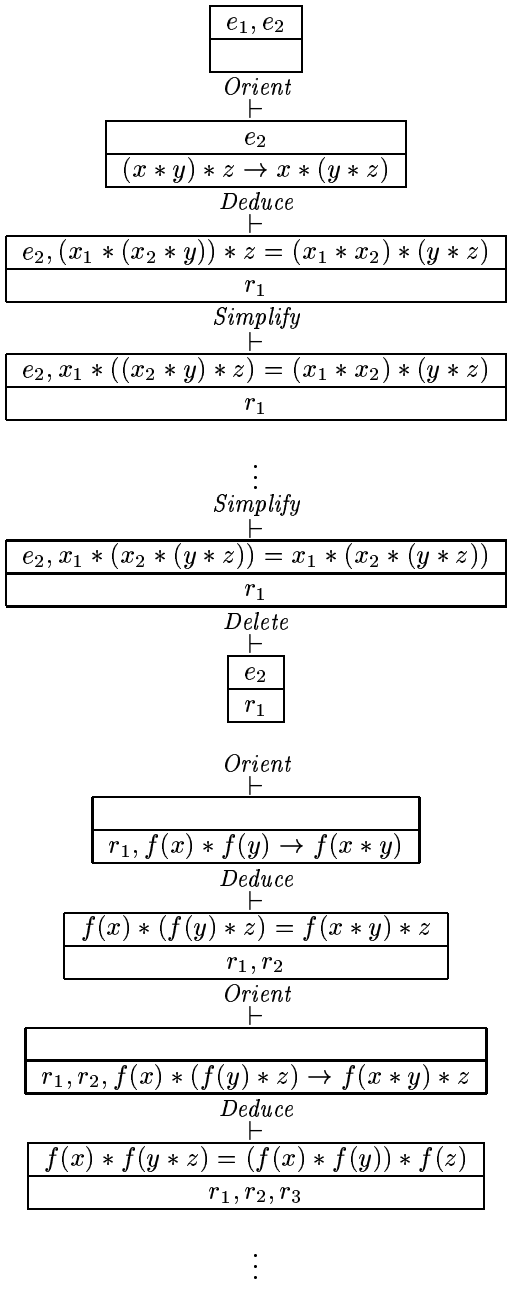
$$E_0 \equiv \{e_1, e_2\}$$

et

$$R_0 \equiv \emptyset,$$

où

$$\begin{array}{lcll} e_1 & \equiv & (x*y)*z & = x*(y*z) \\ e_2 & \equiv & f(x)*f(y) & = f(x*y) \end{array}$$



$$\frac{}{\begin{array}{c}(x*y)*z \to x*(y*z)\\ f(x)*f(y) \to f(x*y)\\ f(x)*(f(y)*z) \to f(x*y)*z\end{array}}\ \textit{Delete}$$

6 Les ensembles R_j et E_j

Une complétion peut-être infinie.

Au cours de son déroulement, la complétion peut créer une infinité d'ensembles R_j et E_j.

On souhaite

- qu'une règle reste indéfiniment dans les R_j seulement si elle "doit" y rester.
- qu'une identité ne reste pas indéfiniment dans les E_j.

m

Cela a une conséquence sur le comportement de la complétion, même si elle finit.

7 De quoi a-t-on besoin pour la complétion ?

des règles de transitions :

- *Orient*,
- *Simplify*,...

un contrôle équitable

une boîte à outils avec

- l'unification,
- le filtrage,
- la réécriture,...

+

une structure de donneés ici (R, E)

8 Notion de persistance

Définition

Une règle r est persistante si elle appartient indéfiniment aux R_j après avoir été créée, c-à-d qu'il existe i tel que

$$r \in \bigcap_{\geq i} R_j.$$

Notation
L'ensemble de règles limite ou ensemble de règles persistantes est

$$R_\infty = \bigcup_{i\geq 0} \bigcap_{j\geq i} R_j$$

9 Équité du contrôle

Définition
Une identité e est persistante si e appartient indéfiniment à l'ensemble E_i's, c-à-d qu'il existe un j tel que $e \in \bigcap_{i\geq j} E_i$.

On note :

$$E_\omega = \bigcup_{i\geq 0} \bigcap_{j\geq i} E_j$$

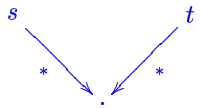
l'ensemble des identités persistantes.

Une paire critique pc est persistante si pc est une paire critique de R_ω.

10 Notion d'équité du contrôle

10.1 Preuve par réécriture

Définition
Une preuve par réécriture est une preuve de la forme:



10.2 Définition de l'équité

Soit R_E le système de réécriture canonique, (c-à-d confluent, noethérien et interréduit) associé à E et $>$.

S'il existe, R_E est unique.

La complétion est équitable si pour toute preuve par réécriture dans R_E, il existe un i tel que cette preuve est une preuve dans R_i.

Autrement dit, le contrôle est équitable si et seulement si

$$R_E = R_\omega$$

10.3 Une condition suffisante d'équité

Un contrôle est équitable si

- Il n'existe pas d'identités persistantes,
- Chaque paire critique persistante est tôt ou tard prise en considération,
- Chaque règle persitante est réduite.

en d'autres termes

$$\begin{array}{rl} E_\omega & = \emptyset \\ cp(R_\omega) & \subset \bigcup_{i\geq 0} E_i = E_\infty \\ R_\omega & \text{est réduit} \end{array}$$

10.4 Un exemple

Il faut considérer toutes les règles, car sinon :

$$\begin{array}{rcl} f(g(h(x))) & \to & g(x) \\ g(h(g(x))) & \to & g(g(h(x))) \\ f(g(g(x))) & \to & f(g(x)) \end{array}$$

Si l'on l'on ne prend jamais en compte la troisième règle,
on obtient les règles $f(g^{n+1}(h(x))) \to g^{n+1}(x)$

Sinon on obtient le système:

$$\begin{array}{rcl} f(g(h(x))) & \to & g(x) \\ g(h(g(x))) & \to & h(x) \\ g(g(x)) & \to & g(x) \end{array}$$

puis

$$\begin{array}{rcl} f(h(x)) & \to & g(x) \\ h(g(x)) & \to & h(x) \\ g(h(x)) & \to & h(x) \\ g(g(x)) & \to & g(x) \\ f(g(x)) & \to & g(x) \end{array}$$

11 La construction d'une bonne complétion

- Puisque les règles de transition sont des bons outils de preuves, pourquoi ne seraient-elles pas de bons outils d'implantation ?
- Une bonne structure de données est cruciale
- La complétion est comme une machine avec la structure de données comme états et les règles de transition comme transitions.
- Le contrôle ou la stratégie nous dit comment invoquer les règles,
- La boîte à outils contient les composants de la procédure, ils peuvent être réutilisés
- L'efficacité est obtenue par des optimisations de haut niveau,
- La lisibilité rend le programme facile à maîtriser,
- Les optimisations de bas niveau sont obtenues au niveau de la boîte à outil,
- Quand on aura obtenu un programme raisonnablement efficace, le programme CAML pourra servir de documentation pour une implantation rapide.

Différents types de complétion ...

12 KB-complétion

12.1 Code

```
let rec KB_Completion ordering (R,E) =

    match E with

        [] -> (R, []) C  (* succes *)

     | (_::_) ->  KB_Completion ordering ((repeat Delete)
                                          ((repeat Simplify)
                                          (Deduction
                                          ((repeat Collapse)
                                          ((repeat Compose)
                                          ((Orient ordering) (R, E)))))))
```

12.2 Figure

E

R

L'implantation de la procédure de Knuth et Bendix

- possède un contrôle très simple et équitable,
- mais elle est inefficace.

Deduction calcule toutes les paires critiques entre les identités de E_j.

13 N-complétion

13.1 Idée

L'implantation de la procédure de Knuth et Bendix

- possède un contrôle très simple et équitable,
- mais elle est inefficace.

La KB-complétion est coûteuse car elle calcule toutes les paires critiques à chaque boucle.

Il est préférable de calculer les paires critiques entre deux règles à la fois et de s'arranger pour ne plus jamais les recalculer.

Cela nécessite la structure de données suivante:

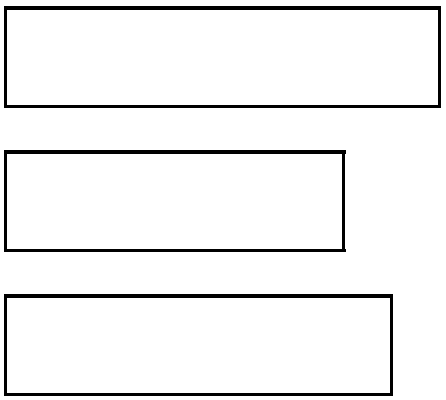
- E est un ensemble d'identités, ou bien paires critiques ou bien identités données au départ.
- T est un ensemble de règles, les règles non marquées dans la terminologie de Huet,
- R est un ensemble de règles dont les paires critiques ont été calculées, les règles marquées

13.2 Code

```
let rec N_Completion ordering (R,T,E) = let COMP = N_Completion ordering in
match (T,E) with
    [],[] ->  (R,[],[]) C{(* success *)
```

```
|  (_::_),[] -> COMP (Deduction
                      (repeat_list [Collapse_T_by_T;
                                    Collapse_T_by_R;
                                    Collapse_R_by_T;
                                    Collapse_R_by_R;
                                    Compose_T;
                                    Compose_R] (R,T,E)))

|  _,(_::_) -> let (R',T',E') = repeat_list[Delete;
                                            Simplify](R,T,E) in
                (match E' with
                     [] -> COMP(R',T',E')
                   | (_::_) -> COMP(Orient ordering (R',T',E')
                               ? failwith "non orientable equation"));;
```

13.3 Figure



14 La N-complétion

La KB-completion est coûteuse car elle calcule toutes les paires critiques à chaque boucle. Il est préférable de calculer les paires critiques entre deux règles à chaque fois.

Cela nécessite la structure de données suivante:

- E est un ensemble d'identités, ou bien paires critiques ou bien identités données au départ.
- T est un ensemble de règles, les règles non marquées dans la terminologie de Huet,
- R est un ensemble de règles dont les paires critiques ont été calculées, les règles marquées

15 S-complétion

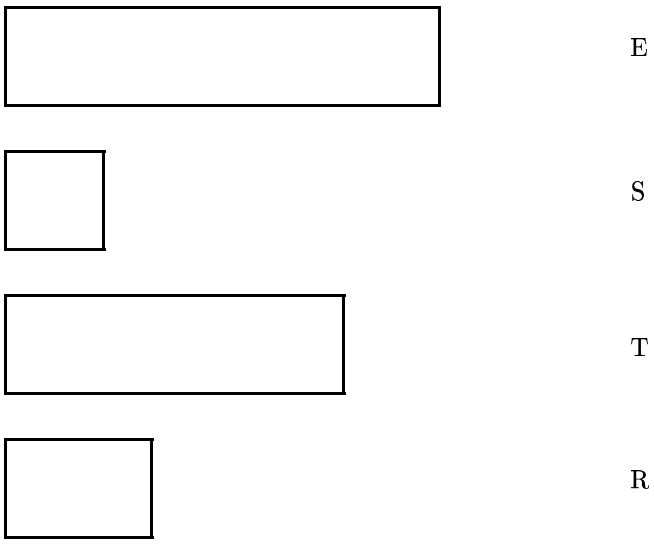
15.1 Structure de données

- E comme dans la N-completion,
- S un ensemble d'identités ordonnées ou règles qui sont utilisés pour simplifier les autres identités ou règles et qui sont appelés des simplifieurs. Pendant la complétion S contient zéro ou une règle,
- T un ensemble de règles déjà utilisées pour simplifier, mais dont les paires critiques n'ont pas encore été calculées.
- R comme dans la N-completion.

15.2 Code

```
 let rec S_Completion ordering (R,T,S,E) =
                      let COMP = S_Completion ordering in
match (T,S,E) with
    [],[],[] ->  (R,[],[],[]) C{(* success *)CE{

 |  _,(_::_),_  -> (COMP (R',T'@S',[],E')
                    where R',T',S',E' =
                         repeat_list [Collapse_T_by_S;
                                      Compose_T_by_S;
                                      Collapse_R_by_S;
                                      Compose_R_by_S] (R,T,S,E))
 |  (_::_),[],[] -> COMP (Deduction (R,T,S,E))

 |  _,[],(_::_) -> let (R',T',S',E') = repeat_list[Delete;
                                                   Simplify]  (R,T,S,E)
                   in (match E' with
                      [] -> COMP(R',T',S',E')
                    | (_::_) -> COMP (Orient ordering (R',T',S',E')
                                ? failwith "non orientable equation"));;
```

15.3 Figure



16 ANS-complétion

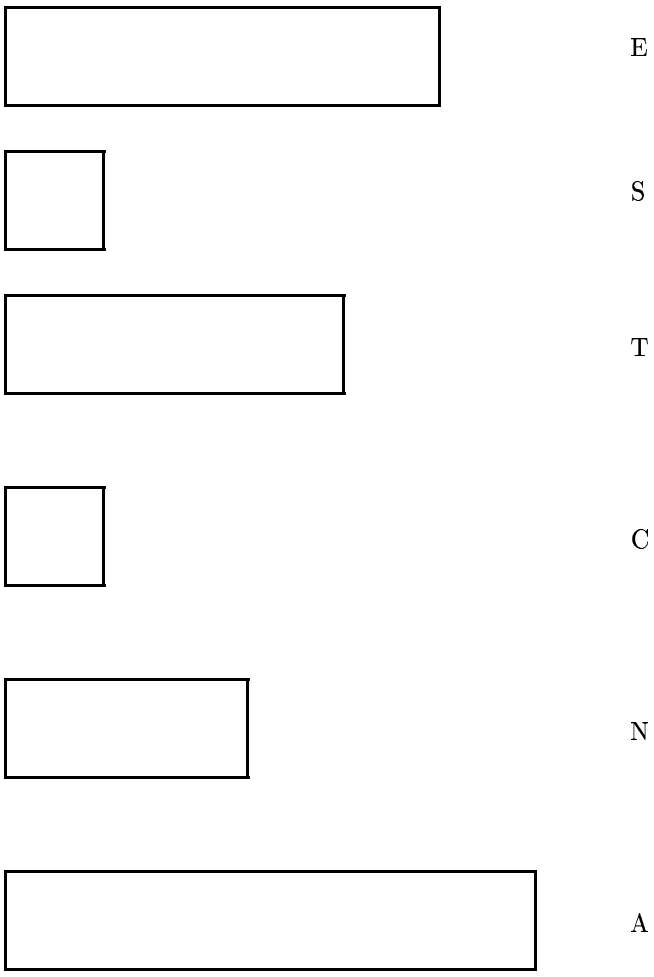
16.1 Structure de données:

- E comme dans la S-completion,
- S comme dans la S-completion,
- T est l'ensemble des règles venant de S et attendant de rentrer dans C.
- C est un ensemble qui contient une ou zéro règle et dont les paires critiques avec les règles de N doivent être calculées.
- N est un ensemble de règles dont les paires critiques n'ont pas été calculées avec les règles de C, mais dont les paires critiques avec les règles de $A \cup N$ ont été calculées.
- A est un ensemble de règles dont les paires critiques avec celle de $A \cup N \cup C$ ont été calculées.
- C est un ensemble qui contient une ou zéro règle et dont les paires critiques avec les règles de N doivent être calculées.
- N est un ensemble de règles dont les paires critiques n'ont pas été calculées avec les règles de C, mais dont les paires critiques avec les règles de $A \cup N$ ont été calculées.
- A est un ensemble de règles dont les paires critiques avec celle de $A \cup N \cup C$ ont été calculées.

16.2 Code

```
let rec ANS_Completion ordering (A,N,C,T,S,E) =
    let COMP = ANS_Completion ordering in

match (N,C,T,S,E) with
   _,[],[],[],[] -> (A,N,C,T,S,E) C{(* success *)

 | _,_,_,(_::_),_ -> (COMP (A',N',C',T'@S',[],E')
            where A',N',C',T',S',E' =
            repeat_list [Collapse_A_by_S;Compose_A_by_S;
                         Collapse_N_by_S;Compose_N_by_S;
                         Collapse_C_by_S;Compose_C_by_S;
                         Collapse_T_by_S;Compose_T_by_S]
                                          (A,N,C,T,S,E))

 | _,_,_,[],(_::_) -> let  A',N',C',T',S',E' =
                                  repeat_list[Delete;
                                              Simplify](A,N,C,T,S,E) in
          (match E' with
              [] ->  COMP(A',N',C',T',S',E')
            | (_::_) ->  COMP(Orient ordering (A',N',C',T',S',E')
                                ? failwith "non orientable equation"))

 | (_::_),[_],_,[],[] -> COMP (Deduction(A,N,C,T,S,E))

 | [],[_],_,[],[] -> COMP (A_C2N crit (Internal_Deduction (A,N,C,T,S,E)))

 | _,[],(_::_),[],[] -> (COMP([],A@N,[r],T',[],[])
                                  where r,T' = least Size T);;
```

16.3 Figure



16.4 Code

17 En résumé ...

La KB-completion est simple et possède un contrôle équitable, mais elle calcule trop de paires critiques,

La N-completion calcule moins de paires critiques, mais ne fait pas assez de simplifications

La S-completion confère aux simplifications une haute priorité,

L'ANS-completion calcule encore moins de paires critiques et fait plus de simplifications,

L'ANIS-completion calcule une paire critique à la fois.

Ces procédures peuvent échouer sur des identités non orientables.

18 STATISTIQUES

Associativité et Endomorphisme ou Distributivité

Procédures	Assoc+Endo			Assoc+Dist		
	filtrages			*filtrages*		
	suc	*échec*	*pc*	*suc*	*échec*	*pc*
KB-completion	16	575	7	19	771	8
N-completion	12	247	5	12	243	5
S-completion	12	168	5	12	148	5
ANS-completion	12	129	5	12	148	5

Groupes

Procédures	Groupes		
	filtrages		
	succès	*échec*	*pc*
KB-completion	1,080	9,540	496
N-completion	220	4,486	109
S-completion	171	3,490	84
ANS-completion	98	1,499	55

Groupes avec division à droite

Procédures	Groupes avec division		
	filtrages		
	succès	*échec*	pc
KB-completion	1300	575,720	647
N-completion	364	11,518	252
S-completion	250	9,719	102
ANS-completion	178	4,967	76

Axiomes de Taussky

Procédures	Taussky		
	filtrages		
	succès	*échec*	pc
KB-completion	-	-	-
N-completion	464	21,987	194
S-completion	445	15,648	183
ANS-completion	324	9,292	139

19 Mesures

Exemples	*Temps*
Distributivité	0.18 s
Groupes	6 s
Naturels	4.4 s
Anneaux booléens	17 s
Groupes abéliens	28 s
Anneaux	150 s

20 les axiomes

Associativité + endomorphisme

$$\begin{aligned} (x*y)*z &= x*(y*z) \\ f(x)*f(y) &= f(x*y) \end{aligned}$$

Associativité + distributivité

$$\begin{aligned} (x+y)+z &= x+(y+z) \\ (x*y)+(x*z) &= x*(y+z) \end{aligned}$$

Groupes

$$\begin{aligned} (x*y)*z &= x*(y*z) \\ e*x &= x \\ i(x)*x &= e \end{aligned}$$

Groupes (système complété)

$$\begin{aligned} (x*y)*z &\to x*(y*z) \\ e*x &\to x \\ x*e &\to x \\ i(x)*x &\to e \\ x*i(x) &\to e \\ i(i(x)) &\to x \\ i(e) &\to e \\ i(x*y) &\to i(y)*i(x) \\ x*(i(x)*y) &\to y \\ i(x)*(x*y) &\to y \end{aligned}$$

Groupes avec divison à droite

$$\begin{aligned} (x*y)*z &= x*(y*z) \\ e*x &= x \\ i(x)*x &= e \\ x*i(y) &= x/y \end{aligned}$$

$$
\begin{array}{rcl}
x * y & \to & x/i(y) \\
x/e & \to & x \\
e/x & \to & i(x) \\
i(e) & \to & e \\
i(i(x)) & \to & x \\
i(x/y) & \to & y/x \\
x/x & \to & e \\
x/(y/z) & \to & (x/i(z))/y \\
(x/y)/i(y) & \to & x \\
(x/i(y))/y & \to & x
\end{array}
$$

Groupes (Axiomes de Taussky)

$$
\begin{array}{rcl}
(x * y) * z & = & x * (y * z) \\
e * e & = & e \\
x * i(x) & = & e \\
g(x * y, y) & = & f(x * y, x) \\
f(e, x) & = & x
\end{array}
$$

Groupes (Axiomes de Taussky, système complété))

$$
\begin{array}{rcl}
(x * y) * z & \to & x * (y * z) \\
e * x & \to & x \\
x * e & \to & x \\
i(x) * x & \to & e \\
x * i(x) & \to & e \\
i(i(x)) & \to & x \\
i(e) & \to & e \\
i(x * y) & \to & i(y) * i(x) \\
x * (i(x) * y) & \to & y \\
i(x) * (x * y) & \to & y \\
g(x, y) & \to & f(x, x * i(y)) \\
f(e, x) & \to & x
\end{array}
$$