

Documentation de `dedunat`

Marc de Falco

13 octobre 2008

1 Fonctionnement

La fenêtre de `dedunat` se compose de 5 parties. De haut en bas :

- le menu, vous permettant en outre de charger ou de sauver une session
- l'affichage des preuves de la session, finies ou en cours, en notation linéaire
- l'affichage des commandes tapées lors de la session
- la boîte de saisie permettant de taper des commandes, on les valide en appuyant sur entrée
- la barre de statut, affichant les messages d'erreur éventuels

2 Notation

Afin de rentrer des symboles on utilise la correspondance suivante, associant un symbole *au clavier* au symbole mathématique correspondant.

\wedge		\wedge
\vee		\vee
\rightarrow		\rightarrow
\sim		\neg
PT		\forall
EX		\exists
$_ _$		\perp
$ -$		\vdash

Lorsqu'une formule est affichée dans la fenêtre des preuves, elle l'est à l'aide des vrais symboles. Cependant, vous pouvez faire du copier-coller sans craintes : ces symboles sont équivalents dans `dedunat` à leur version au clavier. Vous pouvez même les taper directement si vous savez le faire.

3 Commandes

Dans cette section on décrit une par une les commandes que l'on peut effectuer dans `dedunat`. Une commande sera présentée dans une boîte comme ceci :

commande

La fenêtre d'affichage des preuves sera présentée ainsi :

exemple de preuve

3.1 Commencer la preuve d'un séquent

Pour commencer la preuve du séquent $A_1, \dots, A_n \vdash B$ il suffit de taper

$A_1, \dots, A_n \vdash B$

Si il n'y a pas de formules hypothèses dans le séquent on tape

$\vdash B$

voire juste

B

3.2 Commencer la preuve d'une règle dérivée

Afin de prouver la règle dérivée

$$\frac{\Gamma_1 \vdash A_1 \quad \dots \quad \Gamma_n \vdash A_n}{\Gamma \vdash A}$$

on utilise la commande

$\text{Gamma1} \vdash A_1, \dots, \text{Gamman} \vdash A_n \Rightarrow \text{Gamma} \vdash A$

3.3 Abandonner la preuve

Pour abandonner la preuve en cours on tape

quit

3.4 Annuler l'application de la dernière règle

En cas d'erreur, pour annuler la dernière règle appliquée, on tape

```
annuler
```

3.5 Nommer une formule et la remplacer

Afin de nommer une formule pour y faire référence dans une autre formule on utilise la commande

```
soit nom = formule
```

Ce nom est ensuite utilisable dans toute autre formule. Afin de remplacer dans le but courant toutes les occurrences du nom pour la formule qu'il représente on utilise la commande

```
remplacer nom
```

3.6 Donner un synonyme de la conclusion

Afin d'appliquer des règles il peut être nécessaire de donner un synonyme de la conclusion, c'est-à-dire une formule pouvant être comprise différemment par le système mais égale à la conclusion après substitutions et α -conversion. Par exemple pour appliquer une règle (\forall_e) à la conclusion $f(x) = x$ pour quantifier celle-ci sur x , on exprimera cette formule ainsi : $(f(z) = z)[z/x]$, ce qui donnera $\forall x.f(x) = x$ après application de la règle.

La commande à taper est :

```
synonyme formule
```

Lorsqu'on veut juste effectuer des substitutions sur la conclusion, on peut se contenter de taper :

```
simpl
```

Attention cependant, cette commande effectue **toutes** les substitutions possibles.

3.7 Appliquer une règle logique

3.7.1 Axiomes

Si le but actuel est de la forme

$\Gamma, A \vdash A$

la commande

`axiome`

achève sa preuve.

3.7.2 Hypothèses

Dans le cas de la preuve d'une règle dérivée, si le but actuel est de la forme

$\Gamma \vdash A$

où le séquent $\Gamma \vdash A$ fait partie des séquents hypothèses de la règle, la commande

`hypothese`

achève sa preuve.

3.7.3 Règles d'introduction

Les règles d'introductions ont toute la syntaxe

`intro connecteur parametres-eventuels`

On entend par direction le mot `gauche` ou `droite`.

(\rightarrow_i) si le but actuel est de la forme

$\Gamma \vdash A \rightarrow B$

la commande

`intro ->`

donne le résultat

$$\begin{array}{l} \Gamma \vdash A \rightarrow B \quad (\rightarrow_i) \\ \Gamma, A \vdash B \end{array}$$

(\wedge_i) si le but actuel est de la forme

$$\Gamma \vdash A \wedge B$$

la commande

$$\text{intro } \wedge$$

donne le résultat

$$\begin{array}{l} \Gamma \vdash A \wedge B \quad (\wedge_i) : (n)(m) \\ (n) := \Gamma \vdash A \\ (m) := \Gamma \vdash B \\ (n)\Gamma \vdash A \end{array}$$

où n et m sont deux nouveaux numéros de buts. La commande rappelle donc les deux buts avant de commencer la preuve du premier.

$(\vee_i^g), (\vee_i^d)$ si le but actuel est de la forme

$$\Gamma \vdash A \vee B$$

la commande

$$\text{intro } \vee \text{ gauche}$$

donne le résultat

$$\begin{array}{l} \Gamma \vdash A \vee B \quad (\vee_i^g) \\ \Gamma \vdash A \end{array}$$

La règle (\vee_i^d) s'obtient en remplaçant **gauche** par **droite** dans la commande.

(\neg_i) si le but actuel est de la forme

$\Gamma \vdash \neg A$

la commande

`intro ~`

donne le résultat

$\Gamma \vdash \neg A \quad (\neg_i)$
 $\Gamma, A \vdash \perp$

(\forall_i) si le but actuel est de la forme

$\Gamma \vdash \forall x.A$

la commande

`intro PT`

donne le résultat

$\Gamma \vdash \forall x.A \quad (\forall_i)$
 $\Gamma \vdash A$

Attention cette commande n'aboutit que si la règle est applicable, c'est-à-dire que x doit être non libre dans Γ . Pour s'y ramener on peut manuellement effectuer une α -conversion à l'aide de la commande **synonyme**.

(\exists_i) si le but actuel est de la forme

$\Gamma \vdash \exists x.A$

la commande

`intro EX t`

donne le résultat

```
Γ ⊢ ∃x.A      (∃i)
Γ ⊢ A[x\ t]
```

(=)_i) si le but actuel est de la forme

```
Γ ⊢ t = t
```

la commande

```
intro =
```

achève la preuve de ce but.

3.7.4 Règles d'élimination

Les règles d'élimination ont toute la syntaxe

```
elim connecteur parametres-eventuels
```

(→)_e) si le but actuel est de la forme

```
Γ ⊢ B
```

la commande

```
elim -> A
```

donne le résultat

```
Γ ⊢ B      (→e) : (n)(m)
(n) := Γ ⊢ A → B
(m) := Γ ⊢ A
(n)Γ ⊢ A → B
```

(∧^g), (∧^d) si le but actuel est de la forme

$\Gamma \vdash A$

la commande

`elim /\ gauche B`

donne le résultat

$\Gamma \vdash A \quad (\wedge_e^g)$
 $\Gamma \vdash A \wedge B$

De même pour la règle (\wedge_e^d) .

(\vee_e) si le but actuel est de la forme

$\Gamma \vdash C$

la commande

`elim \/ A B`

donne le résultat

$\Gamma \vdash C \quad (\vee_e) : (n)(m)(p)$
 $(n) := \Gamma \vdash A \vee B$
 $(m) := \Gamma, A \vdash C$
 $(p) := \Gamma, B \vdash C$
 $(n)\Gamma \vdash A \vee B$

(\neg_e) si le but actuel est de la forme

$\Gamma \vdash \perp$

la commande

`elim ~ A`

donne le résultat

```
Γ ⊢ ⊥      (¬e) : (n)(m)
(n) := Γ ⊢ A
(m) := Γ ⊢ ¬A
(n)Γ ⊢ A
```

(\perp_i) le faux intuitionniste est considéré ici comme une élimination du faux. Si le but actuel est de la forme

```
Γ ⊢ A
```

la commande

```
elim _|_
```

donne le résultat

```
Γ ⊢ A      (⊥e)
Γ ⊢ ⊥
```

(\forall_e) si le but actuel est de la forme

```
Γ ⊢ A[x\ t]
```

la commande

```
elim PT
```

donne le résultat

```
Γ ⊢ A[x\ t]      (∀e)
Γ ⊢ ∀x.A
```

Pour effectuer directement une élimination sur A en considérant $A = A[x\ x]$ pour obtenir $\forall x.A$, on peut directement taper la commande :

elim PT x

(\exists_e) si le but actuel est de la forme

$\Gamma \vdash C$

la commande

elim EX x A

donne le résultat

$\Gamma \vdash C \quad (\exists_e) : (n)(m)$
 $(n) := \Gamma \vdash \exists x.A$
 $(m) := \Gamma, A \vdash C$
 $(n)\Gamma \vdash \exists x.A$

Attention x doit être non libre dans Γ et C .

$(=_e)$ si le but actuel est de la forme

$\Gamma \vdash A[x \setminus u]$

la commande

elim = v

donne le résultat

$\Gamma \vdash A[x \setminus u] \quad (=_e) : (n)(m)$
 $(n) := \Gamma \vdash A[x \setminus v]$
 $(m) := \Gamma \vdash u = v$
 $(n)\Gamma \vdash A[x \setminus v]$

3.7.5 Affaiblissement

Si le but actuel est de la forme

$\Gamma, B \vdash A$

la commande

affaiblissement B

donne le résultat

$\Gamma \vdash A$

3.7.6 Raisonnement par l'absurde

Si le but actuel est de la forme

$\Gamma \vdash A$

la commande

absurde

donne le résultat

$\Gamma \vdash A \quad (\perp_c)$
 $\Gamma, \neg A \vdash \perp$

4 Exemple

$(A \rightarrow B) \rightarrow A \rightarrow B$

$(0) \vdash (A \rightarrow B) \rightarrow A \rightarrow B$

intro \rightarrow

$(0) \vdash (A \rightarrow B) \rightarrow A \rightarrow B \quad (\rightarrow_i)$
 $A \rightarrow B \vdash A \rightarrow B$

`intro ->`

$(0) \vdash (A \rightarrow B) \rightarrow A \rightarrow B \quad (\rightarrow_i)$
 $A \rightarrow B \vdash A \rightarrow B \quad (\rightarrow_i)$
 $A, A \rightarrow B \vdash B$

`elim -> A`

$(0) \vdash (A \rightarrow B) \rightarrow A \rightarrow B \quad (\rightarrow_i)$
 $A \rightarrow B \vdash A \rightarrow B \quad (\rightarrow_i)$
 $A, A \rightarrow B \vdash B \quad (\rightarrow_e) : (1)(2)$
 $(1) := A, A \rightarrow B \vdash A \rightarrow B$
 $(2) := A, A \rightarrow B \vdash A$
 $(1)A, A \rightarrow B \vdash A \rightarrow B$

`axiome`

$(0) \vdash (A \rightarrow B) \rightarrow A \rightarrow B \quad (\rightarrow_i)$
 $A \rightarrow B \vdash A \rightarrow B \quad (\rightarrow_i)$
 $A, A \rightarrow B \vdash B \quad (\rightarrow_e) : (1)(2)$
 $(1) := A, A \rightarrow B \vdash A \rightarrow B$
 $(2) := A, A \rightarrow B \vdash A$
 $(1)A, A \rightarrow B \vdash A \rightarrow B \quad (ax)$
 $(2)A, A \rightarrow B \vdash A$

`axiome`

$(0) \vdash (A \rightarrow B) \rightarrow A \rightarrow B \quad (\rightarrow_i)$
 $A \rightarrow B \vdash A \rightarrow B \quad (\rightarrow_i)$
 $A, A \rightarrow B \vdash B \quad (\rightarrow_e) : (1)(2)$
 $(1) := A, A \rightarrow B \vdash A \rightarrow B$
 $(2) := A, A \rightarrow B \vdash A$
 $(1)A, A \rightarrow B \vdash A \rightarrow B \quad (ax)$
 $(2)A, A \rightarrow B \vdash A \quad (ax)$
CQFD

5 Exercices

De nombreuses sessions d'exercice sont fournies avec `dedunat`. Toutes sont fournies avec une session corrigé qui contient une preuve du résultat attendu. Il est recommandé de ne regarder le corrigé qu'en cas de réel blocage. Si vous n'arrivez pas à continuer une preuve, prenez une feuille de brouillon pour essayer *à la main* avant de regarder le corrigé.