Lessons from Thérèse's Work

Fairouz Kamareddine (Université de Heriot-Watt, Edimbourgh)

7 Juillet 2010

First Encounter

- I first met Thérèse in 1992 in the Netherlands at a meeting organised by Jan Willem Klop.
- At that time Thérèse had done influential work on systems of rewriting and explicit substitutions.
- I did not know then that she would influence me in many ways (workwise and personality wise).
- In those 18 years I have discovered a deep thinker who is an excellent and a leading academic, and a beautiful human being.
- In this talk I describe how she influenced my academic work in 3 aspects: explicit substitution and rewriting, reduction modulo (versus deduction modulo) and MathLang (cousin-project of Focal).

Le lambda calcul

- Rappelons l'essence du λ-calcul:
  Syntax: $\Lambda ::= \mathcal{V}|(\Lambda\Lambda)|\lambda\mathcal{V}.\Lambda.$
  Règle: $(\lambda x.A)B \rightarrow_\beta A[x := B]$

- La substitution:
  - $x[x := A] = A$
  - $y[x := A] = A$ si $x \neq y$
  - $BC[x := A] = B[x := A]C[x := A]$
  - $(\lambda y.B)[x := A] = \lambda y.B[x := A]$

Le lambda Calcul à la de Bruijn (item notation) [??]

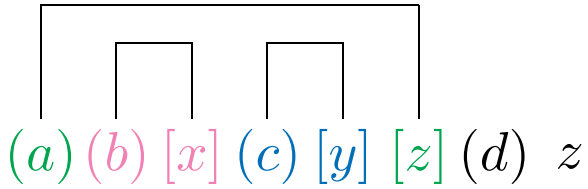
$$
\begin{array}{rcc}
' : \text{ Notation classique} & \mapsto & \text{Notation de de Bruijn} \\
x & \mapsto & x \\
\lambda x.B & \mapsto & [x]B' \\
AB & \mapsto & (B')A'
\end{array}
$$

$$
\text{Example: } (\lambda x.\lambda y.xy)z \quad \mapsto \quad (z)[x][y](y)x
$$

- Dans le *train* $(z)[x][y](y)$, les *wagons* sont (z), $[x]$, $[y]$ and (y).
- Le dernier x dans $(z)[x][y](y)x$ est le *coeur* du terme.
- Le *wagon d'application* (z) et le *wagon d'abstraction* $[x]$ se trouvent l'un *à coté* de l'autre.
- La règle β $\quad (\lambda x.A)B \to_\beta A[x := B]$ devient: $(B)[x]A \to_\beta [x := B]A$

Réduction dans la notation de de Bruijn

Notation classique	Notation de de Bruijn
$\underline{((\lambda_x.(\lambda_y.\lambda_z.zd)c)b)a}$	$(a)\underline{(b)[x]}(c)[y][z](d)z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{((\lambda_y.\lambda_z.zd)c)a}$	$(a)\underline{(c)[y]}[z](d)z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{(\lambda_z.zd)a}$	$\underline{(a)[z]}(d)z$
$\downarrow_\beta$	$\downarrow_\beta$
ad	$(d)a$



Chaque wagon a un *partenaire* sauf (d) qui est *célibataire*.
Les crochets de $((\lambda_x.(\lambda_y.\lambda_z.\ -\ -)c)b)a)$, sont '$[_1\ [_2\ [_3\ ]_2\ ]_1\ ]_3$', où '$[_i$' and '$]_i$' vont ensemble.
Les crochets de $(a)(b)[x](c)[y][z]$ sont plus simple: $[[\,][\,]]$.

Réductions generalisées

- $((\lambda_x.(\underline{\lambda_y.\lambda_z.zd)c})b)a \rightarrow_\beta ((\lambda_x.\{\lambda_z.zd\}[y := c])b)a$

  $(a)(b)[x](c)[y][z](d)z \rightarrow_\beta (a)(b)[x][y := c][z](d)z$

- $(\underline{(\lambda_x.(\lambda_y.\lambda_z.zd)c)b})a \rightarrow_\beta \{(\lambda_y.\lambda_z.zd)c\}[x := b]a$

  $(a)(b)[x](c)[y][z](d)z \rightarrow_\beta (a)[x:=b](c)[y][z](d)z$

- $(a)(b)[x](c)[y][z](d)z \hookrightarrow_\beta (b)[x](c)[y][z := a](d)z$

Quelques notions de réduction dans la literature

Name	In Classical Notation	In de Bruijn's notation
(θ)	$((\lambda_x.N)P)Q$ $\downarrow$ $(\lambda_x.NQ)P$	$(Q)(P)[x]N$ $\downarrow$ $(P)[x](Q)N$
(γ)	$(\lambda_x.\lambda_y.N)P$ $\downarrow$ $\lambda_y.(\lambda_x.N)P$	$(P)[x][y]N$ $\downarrow$ $[y](P)[x]N$
(γ_C)	$((\lambda_x.\lambda_y.N)P)Q$ $\downarrow$ $(\lambda_y.(\lambda_x.N)P)Q$	$(Q)(P)[x][y]N$ $\downarrow$ $(Q)[y](P)[x]N$
(g)	$((\lambda_x.\lambda_y.N)P)Q$ $\downarrow$ $(\lambda_x.N[y := Q])P$	$(Q)(P)[x][y]N$ $\downarrow$ $(P)[x][y := Q]N$

Quelques usages de ces réductions/"term reshuffling"

- **?** introduit θ and γ et les utilise pour analyzer la strategy perpetuelle de réduction.
- [**??**] utilisent term reshuffling pour analyzer des problèmes de typages (comme (non)décidibilité).
- [**???**] les utitlisent pour réduire le problème de normalisation forte à celui de normalisation faible.
- [**?**] montre que term-reshuffling peut bien representer l'évaluation fonctionnelle paresseuse.
- [**????**] montrent que les réductions géneralisés sont plus efficaces (espace et temps).
- [**?**] établit le théorème de conservation pour les réductions generalisées.

Les indices de de Bruijn

- λ-calcul classique: $A ::= x \mid (\lambda x.B) \mid (BC)$
  $(\lambda x.A)B \to_\beta A[x := B]$
- $(\lambda x.\lambda y.xy)y \to_\beta (\lambda y.xy)[x := y] \neq \lambda y.yy$
- $(\lambda x.\lambda y.xy)y \to_\beta (\lambda y.xy)[x := y] =_\alpha (\lambda z.xz)[x := y] = \lambda z.yz$
- $\lambda x.x$ and $\lambda y.y$ sont la même fonction. Soit $\lambda 1$.
- Supposons une liste de variables libres (disons $x, y, z, \ldots$).
- $(\lambda\lambda 2\ 1)2 \to_\beta (\lambda 2\ 1)[1 := 2] = \lambda(2[2 := 3])(1[2 := 3]) = \lambda 3\ 1$

Le λ-calcul classique avec les indices de de Bruijn

- Soient $i, n \geq 1$ et $k \geq 0$
- $A \ ::= \ n \mid (\lambda B) \mid (BC)$
  $(\lambda A)B \rightarrow_\beta A\{\!\{1 \leftarrow B\}\!\}$
- $$\begin{array}{l} U_k^i(AB) = U_k^i(A)\, U_k^i(B) \\ U_k^i(\lambda A) = \lambda(U_{k+1}^i(A)) \end{array} \qquad U_k^i(\mathtt{n}) = \begin{cases} \mathtt{n+i-1} & \text{si} \quad n > k \\ \mathtt{n} & \text{si} \quad n \leq k\,. \end{cases}$$
- $$\begin{array}{lcl} (A_1 A_2)\{\!\{\mathtt{i} \leftarrow B\}\!\} & = & (A_1\{\!\{\mathtt{i} \leftarrow B\}\!\})\,(A_2\{\!\{\mathtt{i} \leftarrow B\}\!\}) \\ (\lambda A)\{\!\{\mathtt{i} \leftarrow B\}\!\} & = & \lambda(A\{\!\{\mathtt{i+1} \leftarrow B\}\!\}) \end{array}$$
  $$\mathtt{n}\{\!\{\mathtt{i} \leftarrow B\}\!\} = \begin{cases} \mathtt{n-1} & \text{si} \quad n > i \\ U_0^i(B) & \text{si} \quad n = i \\ \mathtt{n} & \text{si} \quad n < i\,. \end{cases}$$
- Plusieures implantations des systèmes de preuves et des languages de programmation sont basées sur les indices de de Bruijn.

Using () everywhere

- We will replace (a) by $(a\delta)$.
- We will replace $[x]$ by (λ_x) or $(\varepsilon\lambda_x)$; and $[x : A]$ by $(A\lambda_x)$.
- New items: substitution items $(A\sigma_x)$ and typing items $(\Gamma\tau)$.
- For example:

  (β) $\quad (\lambda_x.A)C \to_\beta A[x := C]$

  (β) $\quad (C\delta)(\lambda_x)A \to_\beta (C\sigma_x)A$

- I was doing this work in 1992 in Holland with Rob Nederpelt when I met Thérèse and learned about her work on explicit substitutions.
- When I had funding to fill with a Postdoc, I chose Thérèse's PhD student Dr Alejandro Ríos to be my postdoctoral fellow started in 1993.

Les wagons de substitutions [?]

- $(B)[x]A \to_\beta [x := B]A$ devient $(B\delta)(\lambda_x)A \to (B\sigma^x)A$
- Si on utilise les indices de de Bruijn, ca devient: $(B\delta)(\lambda)A \to (B\sigma^1)A$
- Les σ-wagons propagent les λ- et δ-wagons:

  $\sigma\delta$-*transition* $\quad (C\sigma^i)(B\delta)A \to ((C\sigma^i)B\delta)(C\sigma^i)A$

  $\sigma\lambda$-*transition* $\quad (C\sigma^i)(\lambda)A \to (\lambda)(C\sigma^{i+1})A$

- On a aussi besoin des φ-wagons qui changent les variables. Ces wagons, eux aussi propagent les λ- et δ-wagons:

  $\varphi\delta$-*transition* $\quad (\varphi^i_k)(B\delta)A \to ((\varphi^i_k)B\delta)(\varphi^i_k)A$

  $\varphi\lambda$-*transition* $\quad (\varphi^i_k)(\lambda)A \to (\lambda)(\varphi^i_{k+1})A$

Les wagons de substitutions [?]

- Il y avait aussi les règles de destruction des nouveaux wagons.

  - $(\varphi_k^i)n \quad \rightarrow \quad \ldots$
  - $(A\sigma^i)n \quad \rightarrow \quad \ldots$

  Et des règles permettant un σ-wagon (resp., un δ-wagon) à propager les σ- et δ-wagons.

  - $\sigma\sigma$-transition $\quad (A\sigma^i)(B\sigma^i)C \quad \rightarrow \quad \ldots$
  - $\sigma\varphi$-transition ...
  - $\varphi\sigma$-transition ...
  - $\varphi\varphi$-transition ...

Le λs_e- calcul [??]

- [**?**] a démontré qu'une restriction λs de ce calcul (qui reflète le calcul lui même de de Bruijn), satisfait les bonnes propriétés sur les termes clos.

- Pour la confluence de λs étendu avec les termes ouverts, [**?**] ajoutait des règles qui reflètent six lemmes sur les propriétés du calcul de de Bruijn donnat λs_e.

From classical λ-calculus with de Bruijn indices to substitution calculus λs [?]

- Ecrivons $A\{\!\{n \leftarrow B\}\!\}$ comme $A\sigma^n\, B$ et $U_k^i(A)$ comme $\varphi_k^i A$.
- $A \;::=\; n \mid (\lambda B) \mid (BC) \mid (A\sigma^i B) \mid (\varphi_k^i B) \qquad where \quad i, n \geq 1\,, \quad k \geq 0\,.$

σ-*generation*	$(\lambda A)\, B$	$\longrightarrow$	$A\, \sigma^1\, B$
σ-λ-*transition*	$(\lambda A)\, \sigma^i B$	$\longrightarrow$	$\lambda(A\, \sigma^{i+1}\, B)$
σ-*app-transition*	$(A_1\, A_2)\, \sigma^i B$	$\longrightarrow$	$(A_1\, \sigma^i B)\, (A_2\, \sigma^i B)$
σ-*destruction*	$\mathtt{n}\, \sigma^i B$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \mathsf{si} \quad n > i \\ \varphi_0^i\, B & \mathsf{si} \quad n = i \\ \mathtt{n} & \mathsf{si} \quad n < i \end{cases}$
φ-λ-*transition*	$\varphi_k^i(\lambda A)$	$\longrightarrow$	$\lambda(\varphi_{k+1}^i\, A)$
φ-*app-transition*	$\varphi_k^i(A_1\, A_2)$	$\longrightarrow$	$(\varphi_k^i\, A_1)\, (\varphi_k^i\, A_2)$
φ-*destruction*	$\varphi_k^i\, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \mathsf{si} \quad n > k \\ \mathtt{n} & \mathsf{si} \quad n \leq k \end{cases}$

Comment on obtient λse de λs?

- $A ::= X \mid n \mid (\lambda B) \mid (BC) \mid (A\sigma^i B) \mid (\varphi_k^i B)$ où $i, n \geq 1\,,\ k \geq 0\,.$
- Ajouter les termes ouverts sans étendre les règles aboutit a la perte de la confluence (même locale):
  $((\lambda X)Y)\sigma^1 1 \to (X\sigma^1 Y)\sigma^1 1 \qquad ((\lambda X)Y)\sigma^1 1 \to ((\lambda X)\sigma^1 1)(Y\sigma^1 1)$
- $(X\sigma^1 Y)\sigma^1 1$ et $((\lambda X)\sigma^1 1)(Y\sigma^1 1)$ n'ont pas un reduct commun.
- Mais , $((\lambda X)\sigma^1 1)(Y\sigma^1 1) \twoheadrightarrow (X\sigma^2 1)\sigma^1(Y\sigma^1 1)$
- La solution est simple: ajoute les lemmes de metasubstitution et distribution aux règles de λs:

σ-σ	$(A\sigma^i B)\,\sigma^j\,C$	$\longrightarrow$	$(A\,\sigma^{j+1}\,C)\,\sigma^i\,(B\,\sigma^{j-i+1}\,C)$	si	$i \leq j$
σ-φ *1*	$(\varphi^i_k\,A)\,\sigma^j\,B$	$\longrightarrow$	$\varphi^{i-1}_k\,A$	si	$k < j < k+i$
σ-φ *2*	$(\varphi^i_k\,A)\,\sigma^j\,B$	$\longrightarrow$	$\varphi^i_k(A\,\sigma^{j-i+1}\,B)$	si	$k+i \leq j$
φ-σ	$\varphi^i_k(A\,\sigma^j\,B)$	$\longrightarrow$	$(\varphi^i_{k+1}\,A)\,\sigma^j\,(\varphi^i_{k+1-j}\,B)$	si	$j \leq k+1$
φ-φ *1*	$\varphi^i_k\,(\varphi^j_l\,A)$	$\longrightarrow$	$\varphi^j_l\,(\varphi^i_{k+1-j}\,A)$	si	$l+j \leq k$
φ-φ *2*	$\varphi^i_k\,(\varphi^j_l\,A)$	$\longrightarrow$	$\varphi^{j+i-1}_l\,A$	si	$l \leq k < l+j$

- Ces nouvelles règles de réecriture sont bien connues: les lemmes de meta-substitution ($\sigma - \sigma$) et de distribution ($\varphi - \sigma$) (et 4 autres règles necessaires pour les demontrer).

D'où venaient ces règles?

Dans le λ-calcul de de Bruijn nous avons les lemmes:

$(\sigma-\varphi\ 1)$ Si $k<j<k+i$ alors: $U_k^{i-1}(A)=U_k^i(A)\{\!\{\mathtt{j}\leftarrow B\}\!\}$.

$(\varphi-\varphi\ 2)$ Si $l\leq k<l+j$ alors: $U_k^i(U_l^j(A))=U_l^{j+i-1}(A)$.

$(\sigma-\varphi\ 2)$ Si $k+i\leq j$ alors: $U_k^i(A)\{\!\{\mathtt{j}\leftarrow B\}\!\}=U_k^i(A\{\!\{\mathtt{j}-\mathtt{i}+\mathtt{1}\leftarrow B\}\!\})$.

$(\sigma-\sigma)$ *[Meta-substitution lemma]* Si $i\leq j$ alors:
$A\{\!\{\mathtt{i}\leftarrow B\}\!\}\{\!\{\mathtt{j}\leftarrow C\}\!\}=A\{\!\{\mathtt{j}+\mathtt{1}\leftarrow C\}\!\}\{\!\{\mathtt{i}\leftarrow B\{\!\{\mathtt{j}-\mathtt{i}+\mathtt{1}\leftarrow C\}\!\}\}\!\}$.

$(\varphi-\varphi\ 1)$ Si $j\leq k+1$ alors:
$U_{k+p}^i(U_p^j(A))=U_p^j(U_{k+p+1-j}^i(A))$.

$(\varphi-\sigma)$ *[Distribution lemma]*
Si $j\leq k+1$ alors: $U_k^i(A\{\!\{\mathtt{j}\leftarrow B\}\!\})=U_{k+1}^i(A)\{\!\{\mathtt{j}\leftarrow U_{k+1-j}^i(B)\}\!\}$.

La preuve de $(\sigma-\sigma)$ utilise $(\sigma-\varphi\ 1)$ et $(\sigma-\varphi\ 2)$ avec $k=0$.
La preuve de $(\sigma-\varphi\ 2)$ utilise $(\varphi-\varphi\ 2)$ avec $l=0$.
Finalement, $(\varphi-\varphi\ 1)$ avec $p=0$ est nécessaire pour démontrer $(\varphi-\sigma)$).

Comment établir la confluence

On utilise la méthode d'interpretation de Thérèse [**?**].

- la méthode d'interpretation de Thérèse a été utilisée pour démontrer la confluence de $\lambda\sigma_{\Uparrow}$-calculus entre autres.
- On croyait qu'on étendait sa technique alors qu'il utilise la normalisation faible en place de la normalisation forte (on ne sais pas si s_e est fortement normalisable).
- Mais, Thérèse avait déja les détails dans sa thèse de doctorat.

What is Thérèse's interpretation method? [?]

- [**?**] l'introduit et l'utilise pour démontrer la confluence de categorical combinators.
- Hardin et Lévy 1989, l'utilise pour démontrer la confluence de $\lambda\sigma_{\Uparrow}$
- In [**?**], it is used to prove the confluence of the weak $\lambda\sigma$-calculus, of the $\lambda\sigma$-calculus on closed terms and the non-confluence of the $\lambda\sigma_{SP}$-calculus on open terms.
- In his PhD, Ríos used it to prove the confluence of the $\lambda\sigma_{SP}$-calculus on semi-closed terms.
- In [**?**] and [**?**] it was used to prove the confluence of $\lambda\upsilon$ and λs respectively.
- In [**?**] it was used to prove the confluence of λs_e.

$\lambda\sigma_{\Uparrow}$

Terms: $\Lambda\sigma^t_{\Uparrow} ::= \mathbb{N} \mid \Lambda\sigma^t_{\Uparrow}\Lambda\sigma^t_{\Uparrow} \mid \lambda\Lambda\sigma^t_{\Uparrow} \mid \Lambda\sigma^t_{\Uparrow}[\Lambda\sigma^s_{\Uparrow}]$

Substitutions: $\Lambda\sigma^s_{\Uparrow} ::= id \mid \uparrow \mid \Uparrow(\Lambda\sigma^s_{\Uparrow}) \mid \Lambda\sigma^t_{\Uparrow} \cdot \Lambda\sigma^s_{\Uparrow} \mid \Lambda\sigma^s_{\Uparrow} \circ \Lambda\sigma^s_{\Uparrow}.$

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\Uparrow(s)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(Varshift1)	$\mathtt{n}\,[\uparrow]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}$
(Varshift2)	$\mathtt{n}\,[\uparrow \circ s]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}\,[s]$
(FVarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(RVarCons)	$\mathtt{n}+\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	$\mathtt{n}\,[s]$
(FVarLift1)	$\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(FVarLift2)	$\mathtt{1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{1}\,[t]$
(RVarLift1)	$\mathtt{n}+\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{n}[s \circ \uparrow]$
(RVarLift2)	$\mathtt{n}+\mathtt{1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{n}[s \circ (\uparrow \circ t)]$

$\lambda\sigma_{\Uparrow}$ rules continued

(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(ShiftLift1)	$\uparrow \circ \Uparrow(s)$	$\longrightarrow$	$s \circ \uparrow$
(ShiftLift2)	$\uparrow \circ (\Uparrow(s) \circ t)$	$\longrightarrow$	$s \circ (\uparrow \circ t)$
(Lift1)	$\Uparrow(s) \circ \Uparrow(t)$	$\longrightarrow$	$\Uparrow(s \circ t)$
(Lift2)	$\Uparrow(s) \circ (\Uparrow(t) \circ u)$	$\longrightarrow$	$\Uparrow(s \circ t) \circ u$
(LiftEnv)	$\Uparrow(s) \circ (a \cdot t)$	$\longrightarrow$	$a \cdot (s \circ t)$
(IdL)	$id \circ s$	$\longrightarrow$	s
(IdR)	$s \circ id$	$\longrightarrow$	s
(LiftId)	$\Uparrow(id)$	$\longrightarrow$	id
(Id)	$a\,[id]$	$\longrightarrow$	a

What is Thérèse's interpretation method? [?]

- Let R be a reduction on A and R^* the reflexive and transitive closure of R.
- Let $R = R_1 \cup R_2$ where R_1 is a confluent and SN reduction on A.
- If there exists a reduction R' on the set of R_1-normal forms satisfying:
  - $R' \subseteq R^*$
  - $a \to_{R_2} b \;\; \Rightarrow \;\; R_1(a) \twoheadrightarrow_{R'} R_1(b)$,
- then R' is confluent iff R is confluent.

Generalised interpretation method (GIM) [?]

- Let $B \subseteq A$, R and R' reduction relations on A and B respectively and $f : A \to B$ such that:
  1. $R' \subseteq R^*$
  2. $\forall a \in A : a \twoheadrightarrow_R f(a)$
  3. $\forall a, b \in A : a \to_R b \;\; \Rightarrow \;\; f(a) \twoheadrightarrow_{R'} f(b)$

- If R' is confluent, then R is also confluent.

- If $4. \forall b \in B : b \twoheadrightarrow_{R'} f(b)$
  then R' is confluent iff R is confluent.

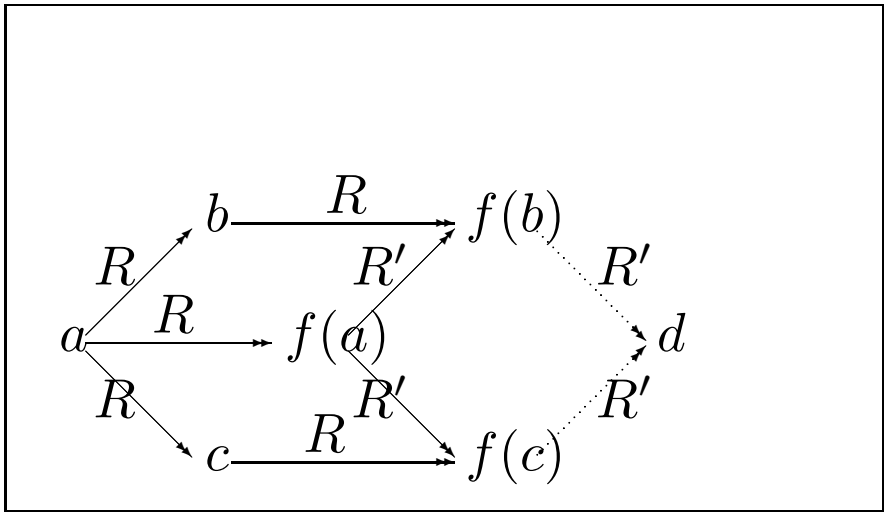


Figure 1: Generalised interpretation method

HOU: What and Why?

- HOU is equational unification for $\beta\eta$-conversion:

$$\text{HOU}\left\{\begin{array}{c}\text{Given two simply-typed lambda terms } a \text{ and } b\\ \text{find a } substitution\ \theta \text{ such that}\\ \theta(a) =_{\beta\eta} \theta(b)\end{array}\right.$$

- HOU is not first order equational unification (FOU) because substitution has to avoid capture of variables.

- Methods of FOU built on grafting cannot be applied to HOU which needs substitution.

- However, HOU can be reduced to FOU in a suitable theory.

Why *making substitutions explicit* is adequate for reasoning about HOU?

- First Order Substitution (Grafting) does not rename variables.
- (Higher Order) Substitution renames variables to avoid capture.
  This complicates the HOU algorithms a lot.
- Grafting cannot replace substitution in λ-calculus:
  Substitution and reduction commute, grafting and reduction do not commute.
- $((\lambda x.Y)a)$ reduces to Y but
  grafting x for Y gives $((\lambda x.x)a)$ which reduces to a and not x.

Why *making substitutions explicit* is adequate for reasoning about HOU?

- Problem comes because of interactions between 2 different calculi: $\beta\eta$-conversion and the application of substitution by the unification algorithm
- The variables that can be meaningfully instantiated by unification are never bound variables (i.e. those that can be instantiated by $\beta\eta$-reduction).
- Hence, distinguish two kinds of variables and set up a calculus where reduction and unification grafting do not interfere.
- In such a calculus, substituting x by a in $((\lambda x.Y)a)$ must be delayed until the unification variable Y is instantiated: Use Explicit Substitutions.

Why *Open Terms*?

- We need two kinds of variables: those of $\beta\eta$-conversion and those of unification.
  Use de Bruijn indices for the first and meta-variables for the second.

- In substitution calculi, grafting and reduction commute.
  So grafting can be used for unification.

- So, with de Bruijn indices, Open terms and explicit substitution:
  a HOU problem in λ-calculus translates into a FOU problem of substitution calculus.

- Solutions of the FOU problem can be translated back as solutions in λ-calculus.

• Terms in de Bruijn notation, $\Lambda_{dB}(\mathcal{X})$: $a ::= \mathbb{N} \mid \mathcal{X} \mid (a\ \ a) \mid \lambda.a$, where $\mathcal{X}$ meta-variables and $\mathbb{N}$ set of de Bruijn indices.

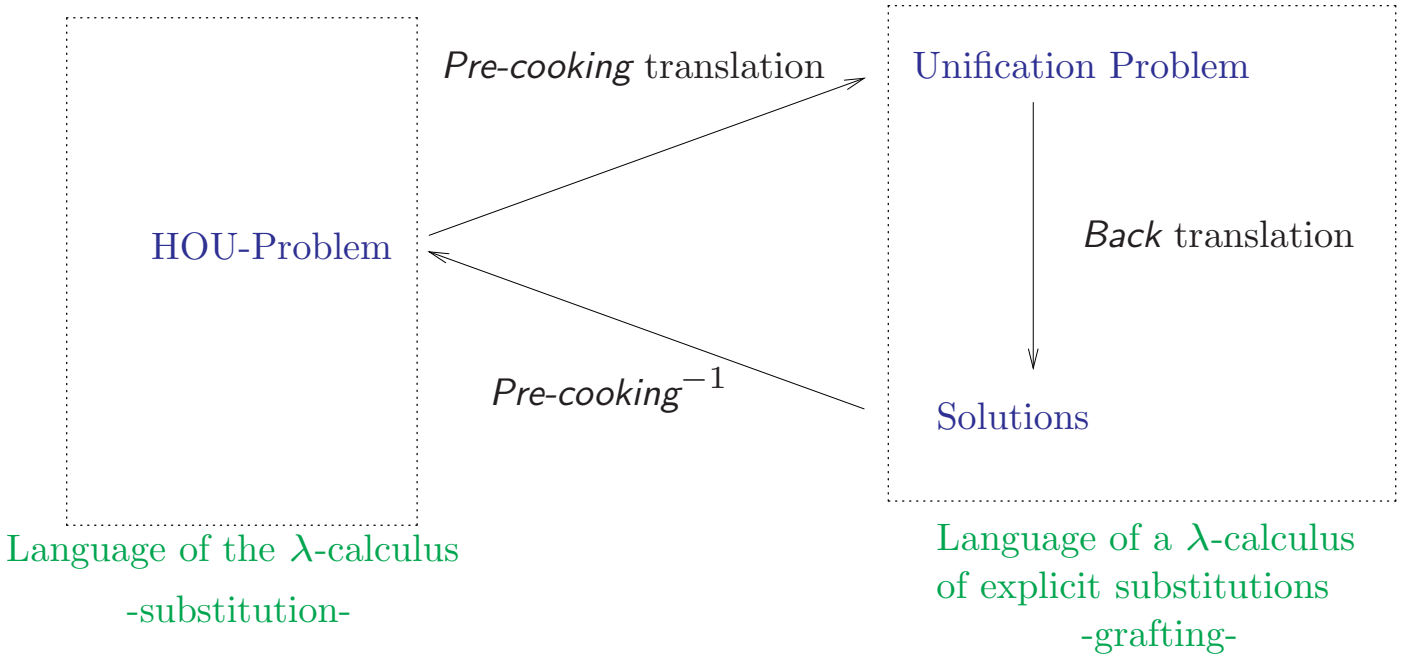
β-reduction: $\lambda.(\lambda.(4\ \ 1)\ \ (\lambda.(2\ \ 1)\ \ 1)) \to_\beta \lambda.(3\ \ (\lambda.(2\ \ 1)\ \ 1)) \to_\beta \lambda.(3\ \ (1\ \ 1))$

• Higher order *substitution*: $\{X/1\}(\lambda.(1\ \ X)\ \ X) = (\lambda.(1\ \ 2)\ \ 1)$

substitution	$\neq$	*grafting*
$\{X/a\}(\lambda.X)$		$(\lambda.X)\{X/a\}$
$\Vert$		$\Vert$
$\lambda.\{X/a^+\}X$		$\lambda.X\{X/a\}$
$\Vert$		$\Vert$
$\lambda.\underbrace{a^+}_{\text{lift}}$	$\neq$	$\lambda.a$

β-reduction

$(\lambda.a\ \ b) \to \{1/b\}a$



- Introduced by G. Dowek, T. Hardin and C. Kirchner using the $\lambda\sigma$-calculus. [**?**]
- Subsumes Huet's HOU method.

Unification in the λs_e-style of explicit substitution [?]

- A λs_e-**unification problem** P is: $\left\{ \bigvee_{j \in J} \underbrace{\exists \vec{w}_j \bigwedge_{i \in I_j} s_i =^?_{\lambda s_e} t_i}_{\text{unification system}} \right.$
- A **unifier** of $\underbrace{\exists \vec{w} \bigwedge_{i \in I} s_i =^?_{\lambda s_e} t_i}_{\text{unification system}}$ is a **grafting** σ such that $\boxed{\exists \vec{w} \bigwedge_{i \in I} s_i \sigma = t_i \sigma}$
- $\vec{w}$ are called **bound** and denoted $\mathcal{B}var(P)$.

Unification rules

Table 1: The Boolean simplification rules for unification problems

(Trivial)	$P \wedge s =^? s$	$\rightarrow$	P
(AndIdem)	$P \wedge e \wedge e$	$\rightarrow$	$P \wedge e$
(OrIdem)	$P \vee e \vee e$	$\rightarrow$	$P \vee e$
(SimpAndT)	$P \wedge \mathbb{T}$	$\rightarrow$	P
(SimpAndF)	$P \wedge \mathbb{F}$	$\rightarrow$	$\mathbb{F}$
(SimpOrT)	$P \vee \mathbb{T}$	$\rightarrow$	$\mathbb{T}$
(SimpOrF)	$P \vee \mathbb{F}$	$\rightarrow$	P
(Distrib)	$P \wedge (Q \vee R)$	$\rightarrow$	$(P \wedge Q) \vee (P \wedge R)$
(Propag)	$\exists \vec{z}(P \vee Q)$	$\rightarrow$	$\exists \vec{z} P \vee \exists \vec{z} Q$
(ElimQE)	$\exists z P$	$\rightarrow$	P, if $z \notin var(P)$
(ElimBV)	$\exists z \quad z =^? t \wedge P$	$\rightarrow$	P, if $z \notin var(P) \cup var(t)$

η-long normal forms

- η-long normal forms guarantee that meta-variables of functional type $A \to B$ are instantiated with typed λs_e-terms of the form $\lambda_A.a$.

- Let a be a λs_e-term of type $A_1 \to \ldots \to A_n \to B$ in the environment Γ and in λs_e-normal form. The η-**long normal form** of a, written a', is defined by:
  1. if $a = \lambda_C.b$ then $a' = \lambda_C.b'$
  2. if $a = (b_1 \ldots b_p)$ then $a' = \lambda_{A_1} \ldots \lambda_{A_n}(c_1 \ldots c_p\, \mathtt{n}' \ldots \mathtt{1}')$, where c_i is the η-long normal form of the normal form of $\varphi_0^{n+1} b_i$
  3. if $a = b\sigma^i c$ then $a' = \lambda_{A_1} \ldots \lambda_{A_n}(d'\sigma^{i+n} e'\, \mathtt{n}' \ldots \mathtt{1}')$, where d', e' are the η-long normal forms of the normal forms of $\varphi_0^{n+1} b$ and $\varphi_0^{n+1} c$, respectively
  4. if $a = \varphi_k^i b$ then $a' = \lambda_{A_1} \ldots \lambda_{A_n}(\varphi_k^i c'\, \mathtt{n}' \ldots \mathtt{1}')$, where c' is the η-long normal form of the normal form of $\varphi_0^{n+1} b$
- The **long normal form** of a λs_e-term is the η-long normal form of its $\beta\eta$-normal form.

A unification system P is a λs_e-**solved form** if all its meta-variables are of atomic type and it is a conjunction of non trivial equations of the following forms:

($Solved$) $X =^?_{\lambda s_e} a$, where the variable X does not occur anywhere else in P and a is in long normal form. Both X and $X =^?_{\lambda s_e} a$ are said to be **solved** in P.

($Flex$-$Flex$) unsolved equations between long normal terms whose leading operator are σ or φ which can be represented as equations between their skeleton: $\psi^{j_p}_{i_p} \dots \psi^{j_1}_{i_1}(X, a_1, \dots, a_p) =^?_{\lambda s_e} \psi^{l_q}_{k_q} \dots \psi^{l_1}_{k_1}(Y, b_1, \dots, b_q)$ with X, Y of atomic type.

Table 2: The λs_e-unification rules

(Dec-λ)	$P \wedge \lambda_A.a =^?_{\lambda s_e} \lambda_A.b \quad \rightarrow \quad P \wedge a =^?_{\lambda s_e} b$
(Dec-App)	$P \wedge (\mathtt{n}\ a_1 \ldots a_p) =^?_{\lambda s_e} (\mathtt{n}\ b_1 \ldots b_p) \quad \rightarrow \quad P \bigwedge_{i=1..p} a_i =^?_{\lambda s_e} b_i$
(App-Fail)	$P \wedge (\mathtt{n}\ a_1 \ldots a_p) =^?_{\lambda s_e} (\mathtt{m}\ b_1 \ldots b_q) \quad \rightarrow \quad \mathbb{F}$ if $\mathtt{n} \neq \mathtt{m}$
(Dec-σ)	$P \wedge a\sigma^i b =^?_{\lambda s_e} c\sigma^i d \quad \rightarrow \quad P \wedge a =^?_{\lambda s_e} c \wedge b =^?_{\lambda s_e} d$
(σ-Fail)	$P \wedge a\sigma^i b =^?_{\lambda s_e} c\sigma^j d \quad \rightarrow \quad \mathbb{F}$
	if $i \neq j$ and $a\sigma^i b =^?_{\lambda s_e} c\sigma^j d$ is not *flex-flex*
(Dec-φ)	$P \wedge \varphi^i_k a =^?_{\lambda s_e} \varphi^i_k b \quad \rightarrow \quad P \wedge a =^?_{\lambda s_e} b$
(φ-Fail)	$P \wedge \varphi^i_k a =^?_{\lambda s_e} \varphi^j_l b \quad \rightarrow \quad \mathbb{F}$
	if $i \neq j$ or $k \neq l$ and $\varphi^i_k a =^?_{\lambda s_e} \varphi^j_l b$ is not *flex-flex*
(Exp-λ)	$P \quad \rightarrow \quad \exists(Y \text{ where } A.\Gamma \vdash Y \ : \ B), P \wedge X =^?_{\lambda s_e} \lambda_A.Y$
	if $(\Gamma \vdash X \ : \ A \rightarrow B) \in \mathcal{T}var(P), Y \notin \mathcal{T}var(P)$, and X is a unsolved variable

Table 3: The λs_e-unification rules

(Exp-App) $P \wedge \psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p) =^?_{\lambda s_e} (\mathtt{m}\ b_1 \dots b_q) \quad \rightarrow$

$$P \wedge \psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p) =^?_{\lambda s_e} (\mathtt{m}\ b_1 \dots b_q) \quad \wedge$$

$$\bigvee_{r \in R_p \cup R_i} \exists H_1, \dots, H_k, X =^?_{\lambda s_e} (\mathtt{r}\ H_1 \dots H_k)$$

- if $\psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p)$ is the skeleton of a λs_e-normal term
- X has an atomic type and is not solved
- $H_1, \dots, H_k$ are variables of appropriate types, not occurring in P, with the environments $\Gamma_{H_i} = \Gamma_X$
- $R_p \subseteq \{i_1, \dots, i_p\}$ of superscripts of the σ operator such that $(\mathtt{r}\ H_1 \dots H_k)$ has the right type $R_i = \begin{cases} \bigcup_{j=0}^{p} \{q\} & \text{if } q > i_{k+1} \\ \emptyset & \text{otherwise} \end{cases}$

and $q = m + p - k - \sum_{l=k+1}^{p} j_l, \quad i_0 = \infty, \quad i_{p+1} = 0$

Table 4: The λs_e-unification rules

(Replace)	$P \wedge X =^?_{\lambda s_e} a \quad \rightarrow \quad \{X/a\}P \wedge X =^?_{\lambda s_e} a$
	if $X \in \mathcal{T}var(P), X \notin \mathcal{T}var(a)$ and $a \in \mathcal{X} \Rightarrow a \in \mathcal{T}var(P)$
(Normalize)	$P \wedge a =^?_{\lambda s_e} b \quad \rightarrow \quad P \wedge a' =^?_{\lambda s_e} b'$
	if a or b is not in long normal form,
	$a' = \begin{cases} \text{the long normal form of } a & \text{if } a \text{ is an unsolved variable} \\ a & \text{otherwise} \end{cases}$
	b' is defined from b similarly to a' from a.

Example :

$$(\lambda.(\lambda.(X\ \ 2)\ \ 1)\ \ Y) \quad =^?_{\lambda se} \quad (\lambda.(Z\ \ 1)\ \ U)$$

$$\downarrow \ X, Z : A \to A; Y, U : A$$

Normalize

$$((X\sigma^2 Y)\sigma^1(\varphi_0^1 Y)\ \ \varphi_0^1 Y) \quad =^?_{\lambda se} \quad (Z\sigma^1 U\ \ \varphi_0^1 U)$$

$$\downarrow$$

Dec-App

$$(X\sigma^2 Y)\sigma^1(\varphi_0^1 Y) =^?_{\lambda se} Z\sigma^1 U \quad \wedge \quad \varphi_0^1 Y =^?_{\lambda se} \varphi_0^1 U$$

$$\downarrow$$

Dec-φ

$$(X\sigma^2 Y)\sigma^1(\varphi_0^1 Y) =^?_{\lambda se} Z\sigma^1 U \quad \wedge \quad Y =^?_{\lambda se} U$$

$$\downarrow$$

Replace

$$(X\sigma^2 Y)\sigma^1(\varphi_0^1 Y) =^?_{\lambda se} Z\sigma^1 Y \quad \wedge \quad Y =^?_{\lambda se} U$$

$$\downarrow_*$$

Exp-λ + Replace

$$((\lambda.X')\sigma^2 Y)\sigma^1(\varphi_0^1 Y) =^?_{\lambda se} (\lambda.Z')\sigma^1 Y \quad \wedge \quad \begin{cases} Y =^?_{\lambda se} U \\ X =^?_{\lambda se} \lambda.X' \\ Z =^?_{\lambda se} \lambda.Z' \end{cases}$$

$$\downarrow_*$$

Normalize + Dec-λ

$$(X'\sigma^3 Y)\sigma^2(\varphi_0^1 Y) =^?_{\lambda se} Z'\sigma^2 Y \quad \wedge \quad \begin{cases} Y =^?_{\lambda se} U \\ X =^?_{\lambda se} \lambda.X' \\ Z =^?_{\lambda se} \lambda.Z' \end{cases}$$

- *Solved* equations:
- *Flex-Flex* equations: $(X'\sigma^3 Y)\sigma^2(\varphi_0^1 Y) =^?_{\lambda s_e} Z'\sigma^2 Y$

$$\left\{ \begin{array}{l} Y =^?_{\lambda s_e} U \\ X =^?_{\lambda s_e} \lambda.X' \\ Z =^?_{\lambda s_e} \lambda.Z' \end{array} \right\} \textit{Solved Forms}$$

• Solutions: $\{Y/X_1, U/X_1\} \bigcup$ solutions for X and Z given by the *Flex-Flex* equation.

Take, for instance, $\{Y/X_1, U/X_1\} \bigcup \{X/\lambda.\mathtt{n}+\mathtt{1}, Z/\lambda.\mathtt{n}\}$ with $n > 2$:
$\underline{(\lambda.(\lambda.(\lambda.\mathtt{n}+\mathtt{1}\ \ \mathtt{2})\ \ \mathtt{1})\ \ X_1)} \rightarrow_\beta (\lambda.(\lambda.\mathtt{n}\ \ \mathtt{2})\ \ X_1) \rightarrow_\beta (\lambda.\mathtt{n}-\mathtt{1}\ \ X_1) \rightarrow_\beta \underline{\mathtt{n}-\mathtt{2}}$
and
$\underline{(\lambda.(\lambda.\mathtt{n}\ \ \mathtt{1})\ \ X_1)} \rightarrow_\beta (\lambda.\mathtt{n}-\mathtt{1}\ \ X_1) \rightarrow_\beta \underline{\mathtt{n}-\mathtt{2}}$

- Correctness: If P reduces to P' then every unifier of P' is a unifier of P.
- Completeness: If P reduces to P' then every unifier of P is a unifier of P'.

Theorem [Correctness and Completeness]

The λs_e-unification rules are correct and complete.

3. Checking arithmetic constraints (versus shifts and composition in $\lambda\sigma$)

λs_e-calculus and λ-calculus $\longrightarrow$ *Term*, *Substitution* } objects $\lambda\sigma$-calculus

λs_e uses all de Bruijn indices: $\mathbb{N}$

$\lambda\sigma$ uses only $\mathtt{1}$, "shift" and "composition": $\mathtt{n} \equiv \mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{n-1}]$

Exp-App $\lambda\sigma$-unification rule

$$P \wedge X[a_1 \dots a_p.\uparrow^n] =^?_{\lambda\sigma} (\mathtt{m}\, b_1 \dots b_q) \quad \rightarrow$$
$$\bigwedge \left\{ \begin{array}{l} P \\ X[a_1 \dots a_p.\uparrow^n] =^?_{\lambda\sigma} (\mathtt{m}\, b_1 \dots b_q) \\ \bigvee_{r \in R_p \cup R_i} \exists H_1 \dots H_k, X =^?_{\lambda\sigma} (\mathtt{r}\ H_1 \dots H_k) \end{array} \right.$$

X not solved and atomic; $H_1, \dots, H_k$ variables of appropriate types; $\Gamma_{H_i} = \Gamma_X$, $R_p \subseteq \{1, \dots, p\}$ such that $(\mathtt{r}\ H_1 \dots H_k)$ has the right type, $R_i =$ if $m \geq n+1$ then $\{m - n + p\}$ else $\emptyset$

Exp-App λs_e-unification rule

$$P \wedge \psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p) =^?_{\lambda s_e} (\mathtt{m}\ b_1 \dots b_q) \quad \rightarrow$$
$$\wedge \begin{cases} P \\ \psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p) =^?_{\lambda s_e} (\mathtt{m}\ b_1 \dots b_q) \\ \bigvee_{r \in R_p \cup R_i} \exists H_1, \dots, H_k, X =^?_{\lambda s_e} (\mathtt{r}\ H_1 \dots H_k) \end{cases}$$

$\psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p)$ skeleton of a λs_e-normal term; X atomic and not solved; $\Gamma_{H_i} = \Gamma_X$, $R_p \subseteq \{i_1, \dots, i_p\}$ of superscripts of the σ operator such that $(\mathtt{r}\ H_1 \dots H_k)$ has the right type, $R_i = \bigcup_{k=0}^{p}$ if $i_k \geq m + p - k - \sum_{l=k+1}^{p} j_l > i_{k+1}$ then $\{m + p - k - \sum_{l=k+1}^{p} j_l\}$ else $\emptyset$, where $i_0 = \infty, i_{p+1} = 0$

In the $\lambda\sigma$-calculus

$X[a_1 \dots a_p.\ \uparrow^n] =^?_{\lambda\sigma} (\mathtt{m}\ b_1 \dots b_q)$ has solutions of the form:

$$\left(\begin{array}{cc} \mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{r-1}] & \underbrace{H_1 \ \dots \ H_k}_{\text{of appropriate type}} \end{array} \right)$$

$$\mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{r-1}]\ \ [a_1 \dots a_p.\ \uparrow^n] = \begin{cases} a_i, \text{ if } 1 \leq r = i \leq p \\ \mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{r-1-p}]\ \ [\underbrace{\uparrow \circ \cdots \circ \uparrow}_{n}] \ otherwise. \end{cases}$$

In the λs_e-calculus

$\psi_{k_p}^{j_p} \dots \psi_{k_1}^{j_1}(X, a_1, \dots, a_p) =^{?}_{\lambda s_e} (\mathtt{m}\ b_1 \dots b_q)$ solutions of the form:

$$\left(\mathtt{n} \ \underbrace{H_1 \ \dots \ H_k}_{\text{of appropriate type}} \right)$$

such that for some i, $\left[\begin{array}{c} k_{i+1} < n \leq k_i \\ \text{and} \\ n - (p - i) + \sum_{r=i+1}^{p} j_r = m \end{array} \right]$

4. Translations between the pure λ-calculus and the λs_e-calculus

- A unifier of $\lambda.X =_{\beta\eta} \lambda.a$ is not a $\{X/b\}$ such that $b =_{\beta\eta} a$:

$$\{X/b\}(\lambda.X) = \lambda.(\{X/b^+\}X) = \lambda.(X\{X/b^+\}) = \lambda.b^+$$

- The **pre-cooking** of a λ-term in de Bruijn notation into the λs_e-calculus is defined by $\boldsymbol{a_{pc}} = PC(a, 0)$ where $PC(a, n)$ is defined by:

1. $PC(\lambda_B.a, n) = \lambda_B.PC(a, n+1)$
2. $PC((a\ \ b), n) = (PC(a, n)\ \ PC(b, n))$
3. $PC(\mathtt{k}, n) = \mathtt{k}$
4. $PC(X, n) = \begin{cases} \text{if } n = 0 \text{ then } X \\ \text{else } \varphi_0^{n+1} X \end{cases}$

Proposition[Semantics of pre-cooking]

$$\underbrace{(\{X_1/b_1,\ldots,X_p/b_p\}(a))_{pc}}_{\text{Substitution}} \quad = \quad \underbrace{a_{pc}\{X_1/b_{1_{pc}},\ldots,X_p/b_{p_{pc}}\}}_{\text{Grafting}}$$

Proposition[Correspondence between solutions]

$$\exists N_1,\ldots,N_p \quad \underbrace{\{X_1/N_1,\ldots,X_p/N_p\}}_{\text{substitution}}(a) \quad =_{\beta\eta} \quad \underbrace{\{X_1/N_1,\ldots,X_p/N_p\}}_{\text{substitution}}(b)$$

$$\Longleftrightarrow$$

$$\exists M_1,\ldots,M_p \quad a_{pc}\underbrace{\{X_1/M_1,\ldots,X_p/M_p\}}_{\text{grafting}} \quad =_{\lambda s_e} \quad b_{pc}\underbrace{\{X_1/M_1,\ldots,X_p/M_p\}}_{\text{grafting}}$$

5. A simple example

Problem: $\lambda.(X\ \ 2) =^?_{\beta\eta} \lambda.2, \quad 2 : A, \quad X : A \to A$

$$
\begin{array}{ll}
\lambda.(\varphi_0^2(X)\ \ 2) =^?_{\lambda se} \lambda.2 & \to Dec\text{-}\lambda \\
(\varphi_0^2(X)\ \ 2) =^?_{\lambda se} 2 & \to Exp\text{-}\lambda \\
\exists Y (\varphi_0^2(X)\ \ 2) =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.Y & \to Replace \\
\exists Y (\varphi_0^2(\lambda.Y)\ \ 2) =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.Y & \to Normalize \\
\exists Y (\varphi_1^2 Y)\sigma^1 2 =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.Y & \to Exp\text{-}app \\
(\exists Y (\varphi_1^2 Y)\sigma^1 2 =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.Y) \wedge (Y =^?_{\lambda se} 1 \vee Y =^?_{\lambda se} 2) & \to Replace \\
((\varphi_1^2 1)\sigma^1 2 =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.1) \vee ((\varphi_1^2 2)\sigma^1 2 =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.2) & \to Normalize \\
(2 =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.1) \vee (2 =^?_{\lambda se} 2 \wedge X =^?_{\lambda se} \lambda.2) & \equiv \\
(X =^?_{\lambda se} \lambda.1) \vee (X =^?_{\lambda se} \lambda.2) &
\end{array}
$$

Problem: $\boxed{\lambda.(X\ \ 2) =^{?}_{\beta\eta} \lambda.2, \quad 2 : A, \quad X : A \to A}$

Solutions: $\begin{cases} \{X/\lambda.1\} \\ \{X/\lambda.2\} \end{cases}$

Note that we have:

$$\{X/\lambda.1\}(\lambda.(X\ \ 2)) = \lambda.(\{X/(\lambda.1)^{+}\}(X)\ \ 2) = \lambda.(\lambda.1^{+1}\ \ 2) = \lambda.(\lambda.1\ \ 2) =_{\beta} \lambda.2$$

and

$$\{X/\lambda.2\}(\lambda.(X\ \ 2)) = \lambda.(\{X/(\lambda.2)^{+}\}(X)\ \ 2) = \lambda.(\lambda.2^{+1}\ \ 2) = \lambda.(\lambda.3\ \ 2) =_{\beta} \lambda.2$$

- $\lambda\sigma$-(HO)Unification and λs_e-(HO)Unification strategies don't differ.
- Pre-cooking (and back) translations in $\lambda\sigma$ and λs_e differ:
  - A simple selection of the scripts for the operators φ and σ in λs_e corresponds to the manipulation of substitution objects in the $\lambda\sigma$-HOU approach.
  - Use of all de Bruijn indices makes our approach simpler.

Reduction Modulo versus Deduction Modulo

- [?] introduced a very neat and useful idea.
- Deduction modulo is a way to remove computational arguments from proofs by reasoning modulo a congruence on propositions.
- Separate computations and deductions in a clean way.
- Instead of $\dfrac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta}$
- They write $\dfrac{\Gamma \vdash_c P, \Delta \quad \Gamma \vdash_c Q, \Delta}{\Gamma \vdash_c R, \Delta}$ if R and $P \wedge Q$ are congruent ($R =_c P \wedge Q$)

Deduction modulo

- If $P =_c Q$ then $\Gamma \vdash_c P, \Delta$ iff $\Gamma \vdash_c Q, \Delta$ and $\Gamma, P \vdash_c \Delta$ iff $\Gamma, Q \vdash_c \Delta$ and the proofs have the same size.
- For each congruence c, there is a set of axioms $\mathcal{T}$ such that $\mathcal{T}, \Gamma \vdash \Delta$ iff $\Gamma \vdash_c \Delta$
- Hence we have the same theorems in both formalisms but the proofs modulo are shorter because the standard deduction steps performing computations described by c are eliminated.

Reduction modulo

- To do reduction modulo, we need the notation of de Bruijn
- To give you an example why this notation is needed, we take the description of normal forms in a substitution calculus.
- The set of terms, noted Λs , of the λs-calculus is given as follows:

$$\Lambda s ::= \mathbb{N} \mid \Lambda s \Lambda s \mid \lambda \Lambda s \mid \Lambda s \, \sigma^i \Lambda s \mid \varphi_k^i \Lambda s \quad where \quad i \geq 1 \,, \quad k \geq 0 \,.$$

The set of open terms, noted Λs_{op} is given as follows:

$$\Lambda s_{op} ::= \mathbf{V} | \mathbb{N} | \Lambda s_{op} \Lambda s_{op} | \lambda \Lambda s_{op} | \Lambda s_{op} \sigma \Lambda s_{op} | \varphi_k^i \Lambda s_{op} \quad where \quad i \geq 1 \,, \quad k \geq 0$$

s_e-normal forms in classical notation

It is cumbersome to describe s_e-normal forms of open terms. But this description is needed to show the weak normalisation of the s_e-calculus. In classical notation, an open term is an s_e-normal form iff one of the following holds:

- $a \in \mathbf{V} \cup \mathbb{N}$, i.e. a is a variable or a de Bruijn number.
- $a = b\,c$, where b and c are s_e-normal forms.
- $a = \lambda b$, where b is an s_e-normal form.
- $a = b\,\sigma^j c$, where c is an s_e-nf and b is an s_e-nf of the form X, or $d\,\sigma^i e$ with $j < i$, or $\varphi^i_k d$ with $j \leq k$.
- $a = \varphi^i_k b$, where b is an s_e-nf of the form X, or $c\,\sigma^j d$ with $j > k+1$, or $\varphi^j_l c$ with $k < l$.

s_e-normal forms in item notation

- Write $a\,\sigma^i b = (b\,\sigma^i)a$ and $\varphi^i_k a = (\varphi^i_k)a$.
- We call a and b the bodies of these items.
- A *normal $\sigma\varphi$-segment* $\overline{s}$ is a sequence of σ- and φ-items such that every pair of adjacent items in $\overline{s}$ are of the form:

  $(\varphi^i_k)(\varphi^j_l)$ and $k < l$ — $(\varphi^i_k)(b\,\sigma^j)$ and $k < j-1$

  $(b\,\sigma^i)(c\,\sigma^j)$ and $i < j$ — $(b\,\sigma^j)(\varphi^i_k)$ and $j \leq k$.

- The s_e-nf's can be described in item notation by the following syntax:

$$NF ::= \mathbf{V} \mid \mathsf{IN} \mid (NF\,\delta)NF \mid (\lambda)NF \mid \overline{s}\,\mathbf{V}$$

  where $\overline{s}$ is a normal $\sigma\varphi$-segment whose bodies belong to NF.

Formes Canoniques [?]

-

des [ ] célibataires	des ()[ ]-paires	des ()s célibataires	coeur
$[x_1]\dots[x_n]$	$(A_1)[y_1]\dots(A_m)[y_m]$	$(B_1)\dots(B_p)$	x

- Dans [?] et [?]

$$\lambda x_1 \cdots \lambda x_n.(\lambda y_1.(\lambda y_2.\cdots(\lambda y_m.xB_p\cdots B_1)A_m\cdots)A_2)A_1$$

- Par example, la forme canonique de:

$$[x][y](a)[z][x'](b)(c)(d)[y'][z'](e)x$$

est

$$[x][y][x'](a)[z](d)[y'](c)[z'](b)(e)x$$

Comment arriver aux formes canoniques?

For $M \equiv [x][y](a)[z][x'](b)(c)(d)[y'][z'](e)x$:

$\theta(M)$:	bach. $[\,]$s $[x][y]$	$()[\,]$-pairs mixed with bach. $[\,]$s $(a)[z][x'](d)[y'](c)[z']$	bach. $()$s $(b)(e)$	end var x
$\gamma(M)$:	bach. $[\,]$s $[x][y][x']$	$()[\,]$-pairs mixed with bach. $()$s $(a)[z](b)(c)[z'](d)[y']$	bach. $()$s (e)	end var x
$\theta(\gamma(M))$:	bach. $[\,]$s $[x][y][x']$	$()[\,]$-pairs $(a)[z](c)[z'](d)[y']$	bach. $()$s $(b)(e)$	end var x
$\gamma(\theta(M))$:	bach. $[\,]$s $[x][y][x']$	$()[\,]$-pairs $(a)[z](d)[y'](c)[z']$	bach. $()$s $(b)(e)$	end var x

$\to_\theta$ and $\to_\gamma$ sont SN et CR. Alors les θ-nf and γ-nf sont unique.

$\theta(\gamma(A))$ et $\gamma(\theta(A))$ sont les deux en *forme canonique*

Notons que: $\theta(\gamma(A)) =_p \gamma(\theta(A))$ où $\to_p$ est la règle

$$(A_1)[y_1](A_2)[y_2]B \to_p (A_2)[y_2](A_1)[y_1]B \qquad \text{si } y_1 \notin \mathrm{FV}(A_2)$$

Réduction basée sur les classes de formes canoniques, Reduction Modulo [?]

- Définissons $[A] = \{B \mid \theta(\gamma(A)) =_p \theta(\gamma(B))\}$.
- Quand $B \in [A]$,on écrit $B \approx_{\mathtt{equi}} A$.
- $\rightarrow_\theta, \rightarrow_\gamma, =_\gamma, =_\theta, =_p \subset \approx_{\mathtt{equi}} \subset =_\beta$ (inclusions strictes).
- $A \rightsquigarrow_\beta B$ iff $\exists A' \in [A].\exists B' \in [B].\ A' \rightarrow_\beta B'$ (avec compatibilité)
- Si $A \rightsquigarrow_\beta B$ alors $\forall A' \in [A].\forall B' \in [B].\ A' \rightsquigarrow_\beta B'$.
- $\rightarrow_\beta \subset \rightarrow_g \subset \rightsquigarrow_\beta \subset =_\beta = \approx_\beta$.
- $\twoheadrightsquigarrow_\beta$ *est CR:* Si $A \twoheadrightsquigarrow_\beta B$ et $A \twoheadrightsquigarrow_\beta C$, alors $\exists D$: $B \twoheadrightsquigarrow_\beta D$ et $C \twoheadrightsquigarrow_\beta D$.
- Soit $r \in \{\rightarrow_\beta, \rightsquigarrow_\beta\}$. Si $A \in SN_r$ et $A' \in [A]$ alors $A' \in SN_r$.
- $A \in SN_{\rightsquigarrow_\beta}$ ssi $A \in SN_{\rightarrow_\beta}$.

Properties of reduction modulo classes

- *$\rightsquigarrow_\beta$ generalises $\rightarrow_g$ and $\rightarrow_\beta$:* $\rightarrow_\beta \subset \rightarrow_g \subset \rightsquigarrow_\beta \subset =_\beta$.
- *$\approx_\beta$ and $=_\beta$ are equivalent:* $A \approx_\beta B$ iff $A =_\beta B$.
- *$\twoheadrightsquigarrow_\beta$ is Church Rosser:*
  If $A \twoheadrightsquigarrow_\beta B$ and $A \twoheadrightsquigarrow_\beta C$, then for some D: $B \twoheadrightsquigarrow_\beta D$ and $C \twoheadrightsquigarrow_\beta D$.
- *Classes preserve $SN_{\rightarrow_\beta}$:* If $A \in SN_{\rightarrow_\beta}$ and $A' \in [A]$ then $A' \in SN_{\rightarrow_\beta}$.
- *Classes preserve $SN_{\rightsquigarrow_\beta}$:* If $A \in SN_{\rightsquigarrow_\beta}$ and $A' \in [A]$ then $A' \in SN_{\rightsquigarrow_\beta}$.
- *$SN_{\rightarrow_\beta}$ and $SN_{\rightsquigarrow_\beta}$ are equivalent:* $A \in SN_{\rightsquigarrow_\beta}$ iff $A \in SN_{\rightarrow_\beta}$.

Le cube de Barendregt

- Syntax: $A ::= x \mid * \mid \Box \mid AB \mid \lambda x{:}A.B \mid \Pi x{:}A.B$
- Formation rule: $$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2} \qquad \mathsf{si}(s_1, s_2) \in \boldsymbol{R}$$

	Simple	Poly-morphic	Depend-ent	Constr-uctors	Related system	Refs.
$\lambda{\to}$	$(*,*)$				λ^τ	[???]
$\lambda 2$	$(*,*)$	$(\Box,*)$			F	[??]
λP	$(*,*)$		$(*,\Box)$		AUT-QE, LF	[??]
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$	POLYREC	[?]
λP2	$(*,*)$	$(\Box,*)$	$(*,\Box)$			[?]
$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$	Fω	[?]
λP$\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$		
λC	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$	CC	[?]

Les règles de typage

(axiom) $$\langle\rangle \vdash * : \Box$$

(start) $$\frac{\Gamma \vdash A : s \quad x \notin \mathrm{DOM}(\Gamma)}{\Gamma, x{:}A \vdash x : A}$$

(weak) $$\frac{\Gamma \vdash A : B \quad \Gamma \vdash C : s \quad x \notin \mathrm{DOM}(\Gamma)}{\Gamma, x{:}C \vdash A : B}$$

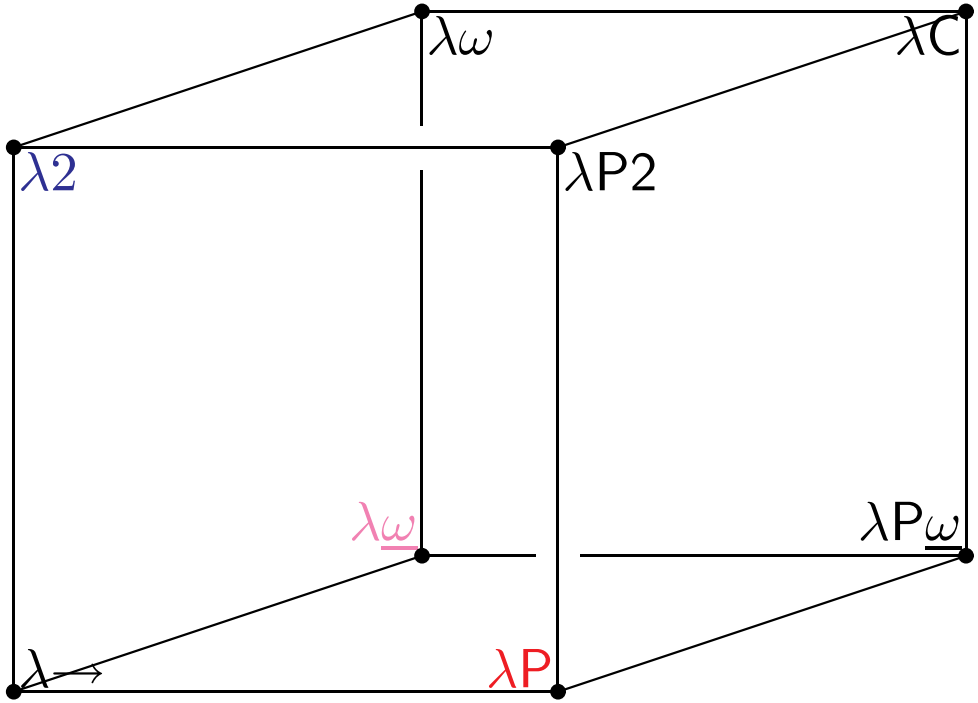
(Π) $$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2 \quad (s_1, s_2) \in \boldsymbol{R}}{\Gamma \vdash \Pi_{x:A}.B : s_2}$$

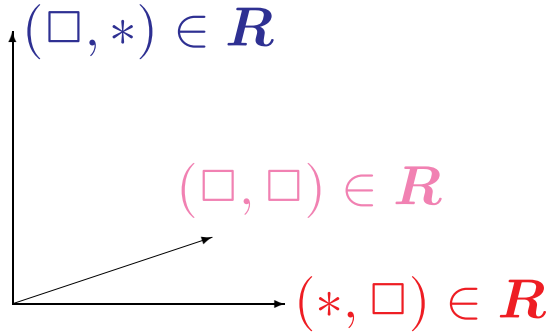
(λ) $$\frac{\Gamma, x{:}A \vdash b : B \quad \Gamma \vdash \Pi_{x:A}.B : s}{\Gamma \vdash \lambda_{x:A}.b : \Pi_{x:A}.B}$$

(conv$_\beta$) $$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash A : B'}$$

(appl) $$\frac{\Gamma \vdash F : \Pi_{x:A}.B \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$$

Le cube de Barendregt





Le cube de Barendregt en notation item avec réduction de classe

- Même formulation sauf que les termes sont écrits en notation item:
- $\mathcal{T} = * \mid \Box \mid V \mid (\mathcal{T}\,\delta)\mathcal{T} \mid (\mathcal{T}\,\lambda_V)\mathcal{T} \mid (\mathcal{T}\,\Pi_V)\mathcal{T}$.
- Les règles de typages ne changent même si on utilise $\rightsquigarrow_\beta$ en place de $\rightarrow_\beta$ (devinez pourquoi).

Le “Subject Reduction” n’ait plus valable

- La plupart des propriétés (inclusif SN) sont valable pour le cube étendu avec la réduction modulo les classes.
- *MAIS* SR n’est pas valable que dans $\lambda_{\rightarrow}$ $(*,*)$ et $\lambda\underline{\omega}$ $(\Box,\Box)$.
- SR ne marche plus dans λP $(*,\Box)$ (et alors dans $\lambda P2, \lambda P\underline{\omega}$ et λC).
- SR ne marche plus dans $\lambda 2$ $(\Box,*)$ (et alors dans $\lambda P2, \lambda\omega$ et λC):

Pourquoi on perd le SR?

- $(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x \rightsquigarrow_\beta (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x$.
- $(\lambda_{\alpha:*}.\lambda_{y:\alpha}.(\lambda_{x:\alpha}.x)y)\beta y' \rightsquigarrow_\beta (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta$
- $\beta : *, y' : \beta \vdash_{\lambda 2} (\lambda_{\alpha:*}.\lambda_{y:\alpha}.(\lambda_{x:\alpha}.x)y)\beta y' : \beta$
- Mais, $\beta : *, y' : \beta \not\vdash_{\lambda 2} (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta : \tau$ pour tout τ.
- On a perrdu l'information que $y' : \beta$ a remplacé $y' : \alpha$ $(\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta$.
- On a besoin de $y' : \alpha$ pour typer le sousterm $(\lambda_{x:\alpha}.x)y'$ de $(\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta$ et alors pour typer $\beta : *, y' : \beta \vdash (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta : \beta$.

Solution de SR: Utilise "let expressions/définitions"

- Définitions/let expressions ont la forme: let $x : A = B$. On les ajoute aux contextes exactement comme les déclarations $y : C$.
- (def rule) $$\frac{\Gamma, \text{let } x : A = B \vdash^{\mathsf{c}} C : D}{\Gamma \vdash^{\mathsf{c}} (\lambda_{x:A}.C)B : D[x := A]}$$
- On définit $\Gamma \vdash^{\mathsf{c}} \cdot =_{\mathtt{def}} \cdot$ comme la relation d' equivalence generée par
  - si $A =_\beta B$ alors $\Gamma \vdash^{\mathsf{c}} A =_{\mathtt{def}} B$
  - si let $x : M = N$ est dans Γ et si B vient de A en substituant une occurrence de x dans A par N, alors $\Gamma \vdash^{\mathsf{c}} A =_{\mathtt{def}} B$.

Le Cube (simplifiée) avec définitions et réduction de classes [?]

(axiom) (app) (abs) et (form) ne changent pas.

(start) $$\frac{\Gamma \vdash^{c} A : s}{\Gamma, x{:}A \vdash^{c} x : A} \qquad \frac{\Gamma \vdash^{c} A : s \quad \Gamma \vdash^{c} B : A}{\Gamma,\ \mathsf{let}\ x : A\ = B \vdash^{c} x : A}$$ x fraic

(weak) $$\frac{\Gamma \vdash^{c} D : E \quad \Gamma \vdash^{c} A : s}{\Gamma, x{:}A \vdash^{c} D : E} \qquad \frac{\Gamma \vdash^{c} A : s \quad \Gamma \vdash^{c} B : A \quad \Gamma \vdash^{c} D : E}{\Gamma,\ \mathsf{let}\ x : A\ = B \vdash^{c} D : E}$$ x fraic

(conv) $$\frac{\Gamma \vdash^{c} A : B \qquad \Gamma \vdash^{c} B' : S \qquad \Gamma \vdash^{c} B =_{\mathtt{def}} B'}{\Gamma \vdash^{c} A : B'}$$

(def) $$\frac{\Gamma, \mathsf{let}\ x : A\ = B \vdash^{c} C : D}{\Gamma \vdash^{c} (\lambda_{x:A}.C)B : D[x := A]}$$

l'usage des définitions résoud le problème de subject reduction

1. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} y' : \beta$
2. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} \alpha =_{\mathtt{def}} \beta$
3. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} y' : \alpha$ (de 1 and 2)
4. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta,\ \mathsf{let}\ x : \alpha = y' \qquad \vdash^{\mathsf{c}} x : \alpha$
5. $\beta : *, y' : \beta,\ \mathsf{let}\ \alpha : * = \beta \qquad \vdash^{\mathsf{c}} (\lambda_{x:\alpha}.x)y' : \alpha[x := y'] = \alpha$

$$\beta : *, y' : \beta \qquad \vdash^{\mathsf{c}} \qquad (\lambda_{\alpha:*}.(\lambda_{x:\alpha}.x)y')\beta : \alpha[\alpha := \beta] = \beta$$

Les propriétés du Cube avec définitions et réduction de classes

- $\vdash^{\mathsf{c}}$ est une generalisation of $\vdash$: Si $\Gamma \vdash A : B$ alors $\Gamma \vdash^{\mathsf{c}} A : B$.
- Equivalent terms have same types:
  Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $A' \in [A]$, $B' \in [B]$ alors $\Gamma \vdash^{\mathsf{c}} A' : B'$.
- Subject Reduction for $\vdash^{\mathsf{c}}$ and $\twoheadrightarrow_\beta$:
  Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $A \twoheadrightarrow_\beta A'$ alors $\Gamma \vdash^{\mathsf{c}} A' : B$.
- Unicity of Types for $\vdash^{\mathsf{c}}$:
  - Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $\Gamma \vdash^{\mathsf{c}} A : B'$ alors $\Gamma \vdash^{\mathsf{c}} B =_{\mathtt{def}} B'$
  - Si $\Gamma \vdash^{\mathsf{c}} A : B$ et $\Gamma \vdash^{\mathsf{c}} A' : B'$ et $\Gamma \vdash^{\mathsf{c}} A =_\beta A'$ alors $\Gamma \vdash^{\mathsf{c}} B =_{\mathtt{def}} B'$.
- Strong Normalisation of $\twoheadrightarrow_\beta$:
  Chaque terme légal est fortement normalisable par rapport à $\twoheadrightarrow_\beta$.

Le language de Mathématique

D'habitude, le mathématicien ignore la logique formelle. Les mathématiciens écrivent la mathématique avec un language (style) commun qu'on appelle le CML. Les avantages de CML:

- *Expressivité:* On peut exprimer toutes genres de notions.
- *Acceptabilité:* CML est accepté par la plupart des mathématicien.
- *traditionalité:* CML existe depuis très longtemps et a été raffiné avec le temps.
- *Universalité:* CML est utilisé partout dans le monde.
- *Flexibilité:* Avec CML on peut décrire plusieures branches de mathématiques.

Les désavantages de CML:

- *Informel et ambigu:* CML est basé sur le language naturelle.
- *Incomplet:* De choses implicites, l'écrivain compte sur l'intuition du lecteur.
- *Pas facile à automatiser* CML
- Au 19ème ciècle, les problèmes en Analyse créaient le besoin d'un style *précis*.
- Plusieures de ces problèmes ont été resolu par le travail de Cauchy (par example par sa définition précise de convergence dans son Cours d'Analyse).
- Les systèmes des nombres sont devenus plus précis avec la définition exacte des nombres réel de Dedekind.
- Cantor commençait la formalisation de la théorie des ensembles et contribuait à la théorie des nombres.

A CML-text

From chapter 1, § 2 of E. Landau's *Foundations of Analysis* (Landau 1930, 1951).

Theorem 6. [Commutative Law of Addition]

$$x + y = y + x.$$

Proof Fix y, and let $\mathfrak{M}$ be the set of all x for which the assertion holds.
I) We have

$$y + 1 = y',$$

and furthermore, by the construction in the proof of Theorem 4,

$$1 + y = y',$$

so that

$$1 + y = y + 1$$

and 1 belongs to $\mathfrak{M}$.
II) If x belongs to $\mathfrak{M}$, then

$$x + y = y + x,$$

Therefore

$$(x + y)' = (y + x)' = y + x'.$$

By the construction in the proof of Theorem 4, we have

$$x' + y = (x + y)',$$

hence

$$x' + y = y + x',$$

so that x' belongs to $\mathfrak{M}$. The assertion therefore holds for all x. □

The problem with formal logic

- No logical language is an alternative to CML
  - A logical language does not have *mathematico-linguistic* categories, is *not universal* to all mathematicians, and is *not a good communication medium*.
  - Logical languages make fixed choices (*first versus higher order, predicative versus impredicative, constructive versus classical, types or sets*, etc.). But different parts of mathematics need different choices and there is no universal agreement as to which is the best formalism.
  - A logician reformulates in logic their *formalization* of a mathematical-text as a formal, complete text which is structured considerably *unlike* the original, and is of little use to the *ordinary* mathematician.
  - Mathematicians do not want to use formal logic and have *for centuries* done mathematics without it.

- *So, mathematicians kept to* CML*.*

- We would like to find an alternative to CML which avoids some of the features of the logical languages which made them unattractive to mathematicians.

What are the options for computerization?

Computers can handle mathematical text at various levels:

- Images of pages may be stored. While useful, this is not a good representation of *language* or *knowledge*.
- Typesetting systems like LaTeX, TeXmacs, can be used.
- Document representations like OpenMath, OMDoc, MathML, can be used.
- Formal logics used by theorem provers (Coq, Isabelle, Mizar, Isar, etc.) can be used.

We are gradually developing a system named Mathlang which we hope will eventually allow building a bridge between the latter 3 levels.

This talk aims at discussing the motivations rather than the details.

The issues with typesetting systems

+ A system like LaTeX, TeXmacs, provides good defaults for visual appearance, while allowing fine control when needed.

+ LaTeX and TeXmacs support commonly needed document structures, while allowing custom structures to be created.

– Unless the mathematician is amazingly disciplined, the *logical structure of symbolic formulas is not represented* at all.

– The *logical structure of mathematics as embedded in natural language text is not represented.* Automated discovery of the semantics of natural language text is still too primitive and requires human oversight.

LaTeX example

draft documents	✓
public documents	✓
computations and proofs	✗

```
\begin{theorem}[Commutative Law of Addition]\label{theorem:6}

  $$x+y=y+x.$$

\end {theorem}

\begin{proof}

  Fix $y$, and $\mathfrak{M}$ be the set of all $x$ for which
  the assertion holds.

  \begin{enumerate}

  \item We have $$y+1=y',$$
    and furthermore, by the construction in
    the proof of Theorem~\ref{theorem:4}, $$1+y=y',$$

    so that $$1+y=y+1$$
    and $1$ belongs to $\mathfrak{M}$.
```

```
  \item If $x$ belongs to $\mathfrak{M}$, then $$x+y=y+x,$$

    Therefore
    $$(x+y)'=(y+x)'=y+x'.$$

    By the construction in the proof of
    Theorem~\ref{theorem:4}, we have $$x'+y=(x+y)',$$

    hence
    $$x'+y=y+x',$$
    so that $x'$ belongs to $\mathfrak{M}$.

  \end{enumerate}

  The assertion therefore holds for all $x$.

\end{proof}
```

The beginnings of computerized formalization

- In 1967 the famous mathematician de Bruijn began work on logical languages for complete books of mathematics that can be *fully* checked by machine.
- People are prone to error, so if a machine can do proof checking, we expect fewer errors.
- Most mathematicians doubted de Bruijn could achieve success, and computer scientists had no interest at all.
- However, he persevered and built *Automath* (AUTOmated MATHematics).
- Today, there is much interest in many approaches to proof checking for verification of computer hardware and software.
- Many theorem provers have been built to mechanically check mathematics and computer science reasoning (e.g. Isabelle, HOL, Coq, Focal, etc.).

Full formalization difficulties: choices

A CML-text is structured differently from a fully formalized text proving the same facts. *Making the latter involves extensive knowledge and many choices:*

- The choice of the *underlying logical system.*
- The choice of *how concepts are implemented* (equational reasoning, equivalences and classes, partial functions, induction, etc.).
- The choice of the *formal foundation*: a type theory (dependent?), a set theory (ZF? FM?), a category theory? etc.
- The choice of the *proof checker*: Automath, Isabelle, Coq, PVS, Mizar, ...

An issue is that one must in general commit to one set of choices.

Full formalization difficulties: informality

Any informal reasoning in a CML-text will cause various problems when fully formalizing it:

- A single (big) step may need to expand into a (series of) syntactic proof expressions. *Very long expressions can replace a clear* CML*-text.*
- The entire CML-text may need *reformulation* in a fully *complete* syntactic formalism where every detail is spelled out. New details may need to be woven throughout the entire text. The text may need to be *turned inside out.*
- Reasoning may be obscured by *proof tactics*, whose meaning is often *ad hoc* and implementation-dependent.

Regardless, ordinary mathematicians do not find the new text useful.

Coq example

draft documents	✗
public documents	✗
computations and proofs	✓

From Module `Arith.Plus` of Coq standard library (`http://coq.inria.fr/`).

```
Lemma plus_sym: (n,m:nat)(n+m)=(m+n).

Proof.

Intros n m ; Elim n ; Simpl_rew ; Auto with arith.

Intros y H ; Elim (plus_n_-Sm m y) ; Simpl_rew ; Auto with arith.

Qed.
```

Mathlang's Goal: Open borders between mathematics, logic and computation

- Ordinary mathematicians *avoid* formal mathematical logic.
- Ordinary mathematicians *avoid* proof checking (via a computer).
- Ordinary mathematicians *may use* a computer for computation: there are over 1 million people who use Mathematica (including linguists, engineers, etc.).
- Mathematicians may also use other computer forms like Maple, LaTeX, etc.
- But we are not interested in only *libraries* or *computation* or *text editing*.
- We want *freedeom of movement* between mathematics, logic and computation.
- At every stage, we must have *the choice* of the level of formalilty and the depth of computation.

Aim for Mathlang? (Kamareddine and Wells 2001, 2002)

Can we formalise a mathematical text, avoiding as much as possible the ambiguities of natural language, while still guaranteeing the following four goals?

1. The formalised text looks very much like the original mathematical text (and hence the content of the original mathematical text is respected).

2. The formalised text can be fully manipulated and searched in ways that respect its mathematical structure and meaning.

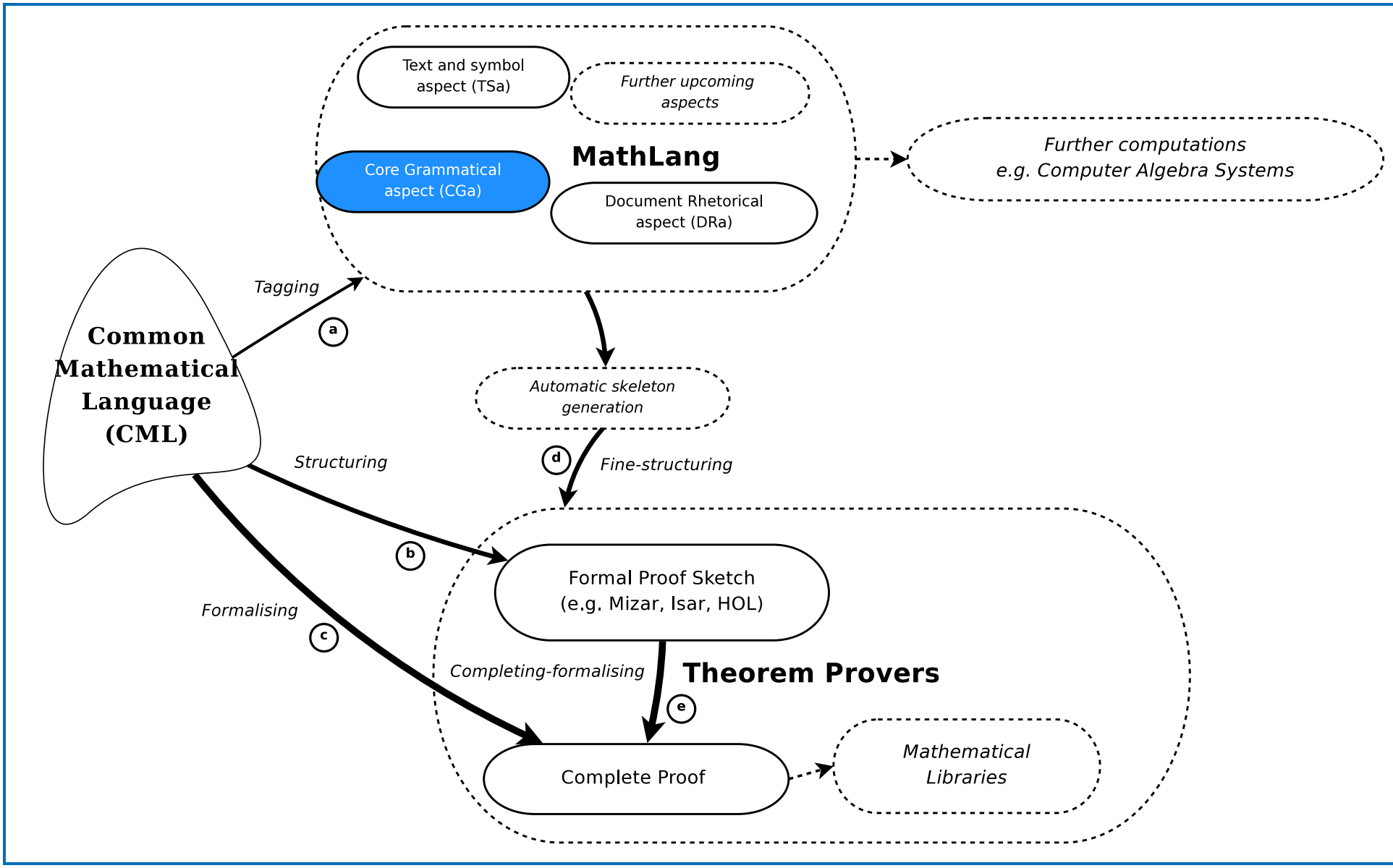
3. Steps can be made to do computation (via computer algebra systems) and proof checking (via proof checkers) on the formalised text.

4. This formalisation of text is not much harder for the ordinary mathematician than LaTeX. *Full formalization down to a foundation of mathematics is not required*, although allowing and supporting this is one goal.

(No theorem prover's language satisfies these goals.)

Mathlang	**draft documents**	✓
	public documents	✓
	computations and proofs	✓

- A Mathlang text captures the grammatical and reasoning aspects of mathematical structure for further computer manipulation.
- A *weak type system* checks Mathlang documents at a grammatical level.
- A Mathlang text remains *close* to its CML original, allowing confidence that the CML has been captured correctly.
- We have been developing ways to weave natural language text into Mathlang.
- Mathlang aims to eventually support *all encoding uses*.
- The CML view of a Mathlang text should match the mathematician's intentions.
- The formal structure should be suitable for various automated uses.


Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

What is CGa? (Maarek's PhD thesis)

- CGa is a formal language derived from MV (N.G. de Bruijn 1987) and WTT (Kamareddine and Nederpelt 2004) which aims at expliciting the grammatical role played by the elements of a CML text.
- The structures and common concepts used in CML are captured by CGa with a finite set of grammatical/linguistic/syntactic categories: *Term* "$\sqrt{2}$", *set* "$\mathbb{Q}$", *noun* "number", *adjective* "even", *statement* "$a = b$", *declaration* "Let a be a number", *definition* "An even number is..", *step* "a is odd, hence $a \neq 0$", *context* "Assume a is even".
- Generally, each syntactic category has a corresponding *weak type*.
- CGa's type system derives typing judgments to check whether the reasoning parts of a document are coherently built.

Examples of linguistic categories

- Terms: the triangle ABC; the center of ABC; $d($x, y$)$.
- Nouns: a triangle; an edge of ABC; a group.
- Adjectives: equilateral triangle; prime number; Abelian group.
- Statements: P lies between Q and R; 5$>$3; AB is an edge of ABC.
- Definition: a number p is prime whenever ....

Weak Type Theory

In Weak Type Theory (or WTT) we have the following linguistic categories:

- On the *atomic* level: *variables*, *constants* and *binders*,
- On the *phrase* level: *terms* $\mathcal{T}$, *sets* $\mathbb{S}$, *nouns* $\mathcal{N}$ and *adjectives* $\mathcal{A}$,
- On the *sentence* level: *statements* P and *definitions* $\mathcal{D}$,
- On the *discourse* level: *contexts* $\mathbb{\Gamma}$, *lines* $\mathbf{l}$ and *books* $\mathbf{B}$.

Categories of syntax of WTT

Other category	abstract syntax	symbol
expressions	$\mathcal{E} = T \vert \mathbb{S} \vert \mathcal{N} \vert P$	E
parameters	$\mathcal{P} = T \vert \mathbb{S} \vert P$ (note: $\vec{\mathcal{P}}$ is a list of $\mathcal{P}$s)	P
typings	$\mathbf{T} = \mathbb{S} :$ SET $\vert \mathcal{S} :$ STAT $\vert T : \mathbb{S} \vert T : \mathcal{N} \vert T : \mathcal{A}$	T
declarations	$\mathcal{Z} = \mathtt{V}^S :$ SET $\vert \mathtt{V}^P :$ STAT $\vert \mathtt{V}^T : \mathbb{S} \vert \mathtt{V}^T : \mathcal{N}$	Z

level	category	abstract syntax	symbol
atomic	*variables*	$\mathtt{V} = \mathtt{V}^T \vert \mathtt{V}^S \vert \mathtt{V}^P$	x
	constants	$\mathtt{C} = \mathtt{C}^T \vert \mathtt{C}^S \vert \mathtt{C}^N \vert \mathtt{C}^A \vert \mathtt{C}^P$	c
	binders	$\mathtt{B} = \mathtt{B}^T \vert \mathtt{B}^S \vert \mathtt{B}^N \vert \mathtt{B}^A \vert \mathtt{B}^P$	b
phrase	*terms*	$T = \mathtt{C}^T(\overrightarrow{\mathcal{P}}) \vert \mathtt{B}^T_{\mathcal{Z}}(\mathcal{E}) \vert \mathtt{V}^T$	t
	sets	$\mathbb{S} = \mathtt{C}^S(\overrightarrow{\mathcal{P}}) \vert \mathtt{B}^S_{\mathcal{Z}}(\mathcal{E}) \vert \mathtt{V}^S$	s
	nouns	$\mathcal{N} = \mathtt{C}^N(\overrightarrow{\mathcal{P}}) \vert \mathtt{B}^N_{\mathcal{Z}}(\mathcal{E}) \vert \mathcal{A}\mathcal{N}$	n
	adjectives	$\mathcal{A} = \mathtt{C}^A(\overrightarrow{\mathcal{P}}) \vert \mathtt{B}^A_{\mathcal{Z}}(\mathcal{E})$	a
sentence	*statements*	$P = \mathtt{C}^P(\overrightarrow{\mathcal{P}}) \vert \mathtt{B}^P_{\mathcal{Z}}(\mathcal{E}) \vert \mathtt{V}^P$	S
	definitions	$\mathcal{D} = \mathcal{D}^\varphi \vert \mathcal{D}^P$	D
		$\mathcal{D}^\varphi = \mathtt{C}^T(\overrightarrow{V}) := T \vert \mathtt{C}^S(\overrightarrow{V}) := \mathbb{S} \vert$ $\mathtt{C}^N(\overrightarrow{V}) := \mathcal{N} \vert \mathtt{C}^A(\overrightarrow{V}) := \mathcal{A}$	
		$\mathcal{D}^P = \mathtt{C}^P(\overrightarrow{V}) := P$	
discourse	*contexts*	$\mathbb{\Gamma} = \emptyset \mid \mathbb{\Gamma}, \mathcal{Z} \mid \mathbb{\Gamma}, P$	Γ
	lines	$\mathbf{l} = \mathbb{\Gamma} \triangleright P \mid \mathbb{\Gamma} \triangleright \mathcal{D}$	l
	books	$\mathbf{B} = \emptyset \mid \mathbf{B} \circ \mathbf{l}$	B

Derivation rules

(1) B is a weakly well-typed book: $\vdash\ B :: \mathbf{B}$.

(2) Γ is a weakly well-typed context relative to book B: $B\ \vdash\ \Gamma :: \mathbb{\Gamma}$.

(3) t is a weakly well-typed term, etc., relative to book B and context Γ:

$$B;\Gamma\ \vdash\ t :: T, \qquad B;\Gamma\ \vdash\ s :: S, \qquad B;\Gamma\ \vdash\ n :: N,$$
$$B;\Gamma\ \vdash\ a :: A, \qquad B;\Gamma\ \vdash\ p :: P, \qquad B;\Gamma\ \vdash\ d :: D$$

$OK(B;\Gamma)$. stands for: $\vdash\ B :: \mathbf{B}$, *and* $B\ \vdash\ \Gamma :: \mathbb{\Gamma}$

Examples of derivation rules

- $\texttt{dvar}(\emptyset) = \emptyset \qquad \texttt{dvar}(\Gamma', x : W) = \texttt{dvar}(\Gamma'), x \qquad \texttt{dvar}(\Gamma', P) = \texttt{dvar}(\Gamma')$

$$\frac{OK(B;\Gamma), \quad x \in \mathtt{V}^{\mathtt{T/S/P}}, \quad x \in \texttt{dvar}(\Gamma)}{B;\Gamma \ \vdash \ x :: T/S/P} \ (var)$$

$$\frac{B;\Gamma \ \vdash \ n :: N, \quad B;\Gamma \ \vdash \ a :: A}{B;\Gamma \ \vdash \ an :: N} \ (adj{-}noun)$$

$$\frac{}{\vdash \ \emptyset :: \mathbf{B}} \ (emp{-}book)$$

$$\frac{B;\Gamma \ \vdash \ p :: P}{\vdash \ B \circ \Gamma \triangleright p :: \mathbf{B}} \qquad \frac{B;\Gamma \ \vdash \ d :: D}{\vdash \ B \circ \Gamma \triangleright d :: \mathbf{B}} \ (book{-}ext)$$

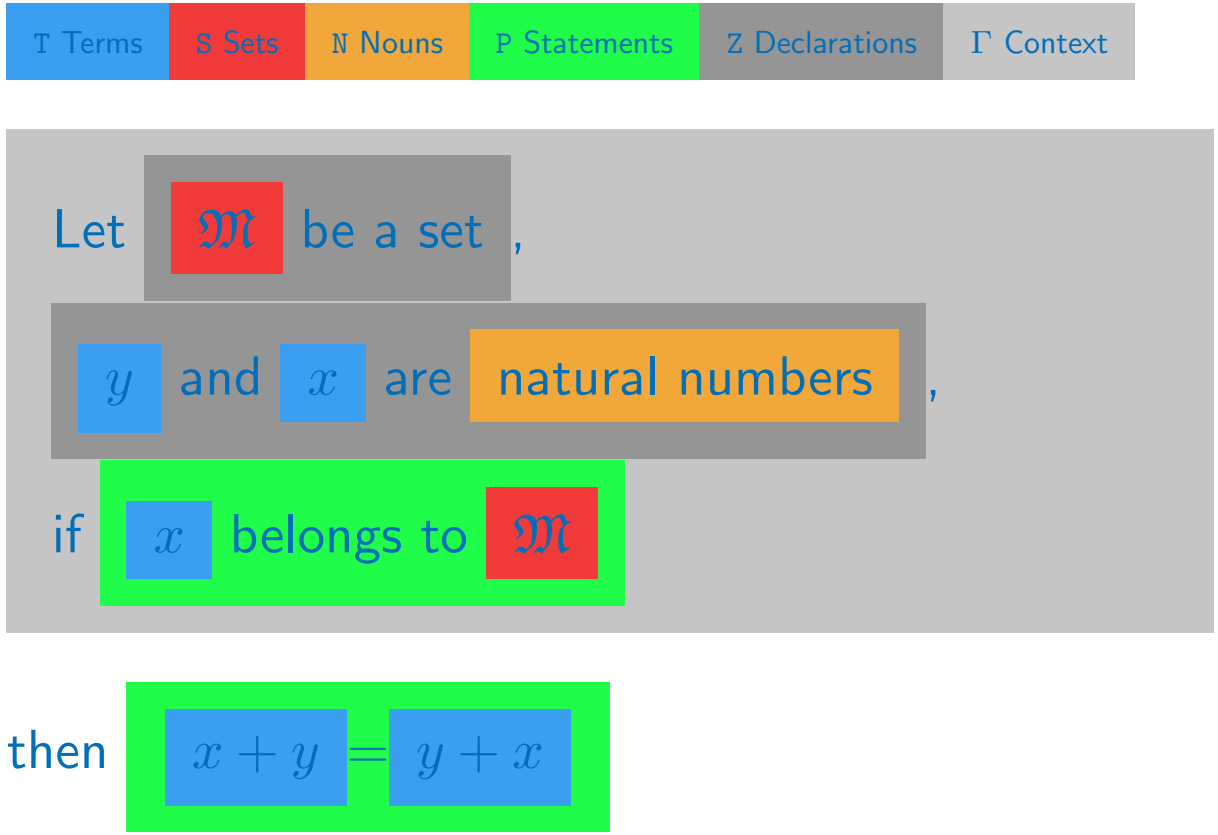
Properties of WTT

- *Every variable is declared* If $B;\Gamma \ \vdash \ \Phi :: \mathbf{W}$ then $FV(\Phi) \subseteq \mathtt{dvar}(\Gamma)$.
- *Correct subcontexts* If $B \ \vdash \ \Gamma :: \mathbb{T}$ and $\Gamma' \subseteq \Gamma$ then $B \ \vdash \ \Gamma' :: \mathbb{T}$.
- *Correct subbooks* If $\ \vdash \ B :: \mathbf{B}$ and $B' \subseteq B$ then $\ \vdash \ B' :: \mathbf{B}$.
- *Free constants are either declared in book or in contexts* If $B;\Gamma \ \vdash \ \Phi :: \mathbf{W}$, then $FC(\Phi) \subseteq \mathtt{prefcons}(B) \cup \mathtt{defcons}(B)$.
- *Types are unique* If $B;\Gamma \ \vdash \ A :: \mathbf{W_1}$ and $B;\Gamma \ \vdash \ A :: \mathbf{W_2}$, then $\mathbf{W_1} \equiv \mathbf{W_2}$.
- *Weak type checking is decidable* there is a decision procedure for the question $B;\Gamma \ \vdash \ \Phi :: \mathbf{W} \ ?$.
- *Weak typability is computable* there is a procedure deciding whether an answer exists for $B;\Gamma \ \vdash \ \Phi :: ?$ and if so, delivering the answer.

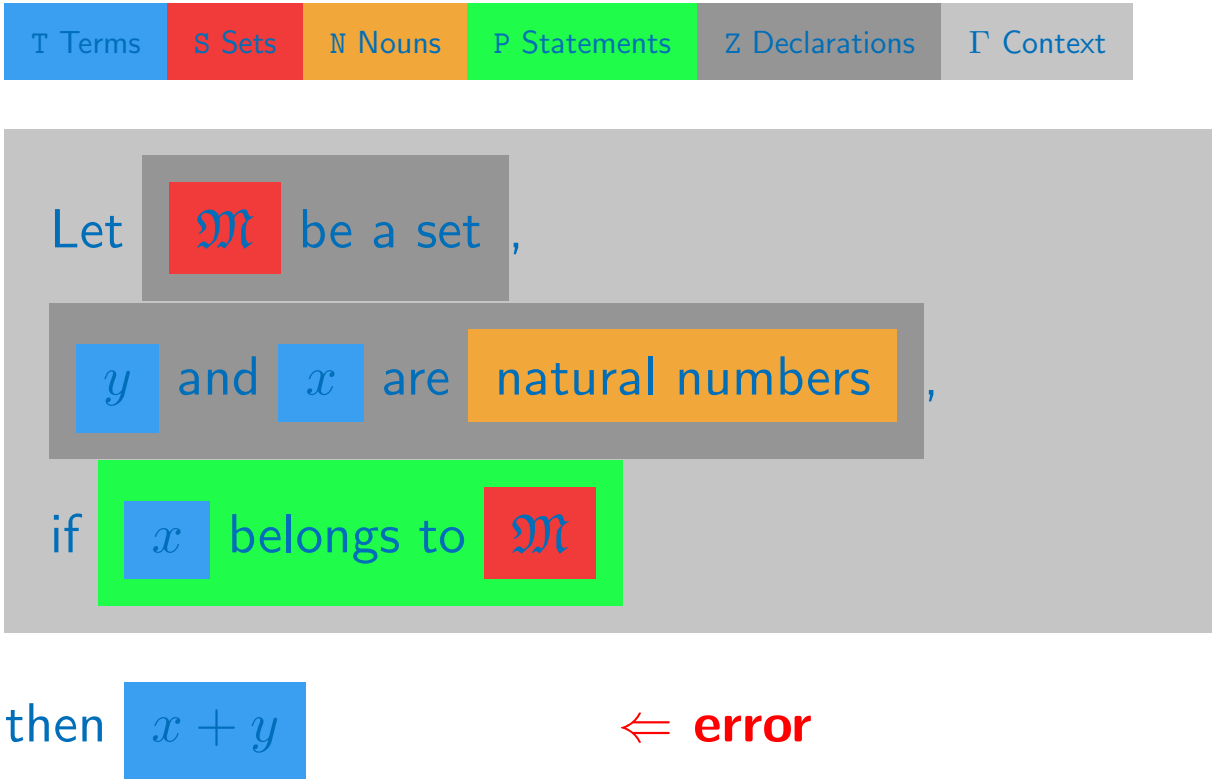
Definition unfolding

- Let $\vdash B :: \mathbf{B}$ and $\Gamma \triangleright c(x_1, \ldots, x_n) := \Phi$ a line in B.
- We write $B \vdash c(P_1, \ldots, P_n) \xrightarrow{\delta} \Phi[x_i := P_i]$.
- *Church-Rosser* If $B \vdash \Phi \xrightarrow{\delta}\!\!\!\!\rightarrow \Phi_1$ and $B \vdash \Phi \xrightarrow{\delta}\!\!\!\!\rightarrow \Phi_2$ then there exists Φ_3 such that $B \vdash \Phi_1 \xrightarrow{\delta}\!\!\!\!\rightarrow \Phi_3$ andf $B \vdash \Phi_2 \xrightarrow{\delta}\!\!\!\!\rightarrow \Phi_3$.
- *Strong Normalisation* Let $\vdash B :: \mathbf{B}$. For all subformulas Ψ occurring in B, relation $\xrightarrow{\delta}$ is strongly normalizing (i.e., definition unfolding inside a well-typed book is a well-founded procedure).

CGa Weak Type Checking



CGa Weak Type checking detects grammatical errors

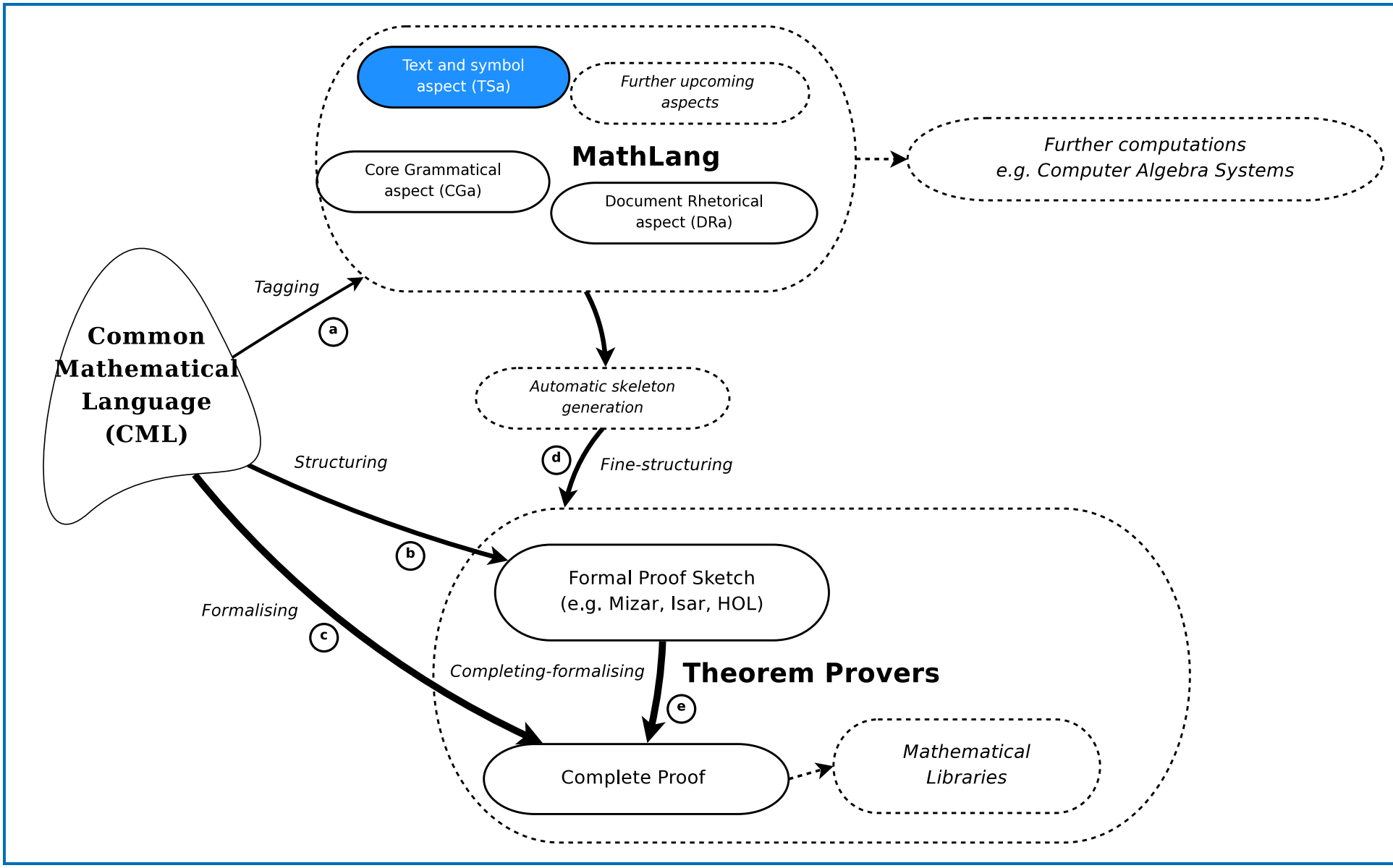


How complete is the CGa?

- CGa is quite advanced but remains under development according to new translations of mathematical texts. Are the current CGa categories sufficient?

- The metatheory of WTT has been established in (Kamareddine and Nederepelt 2004). That of CGa remains to be established. However, since CGa is quite similar to WTT, its metatheory might be similar to that of WTT.

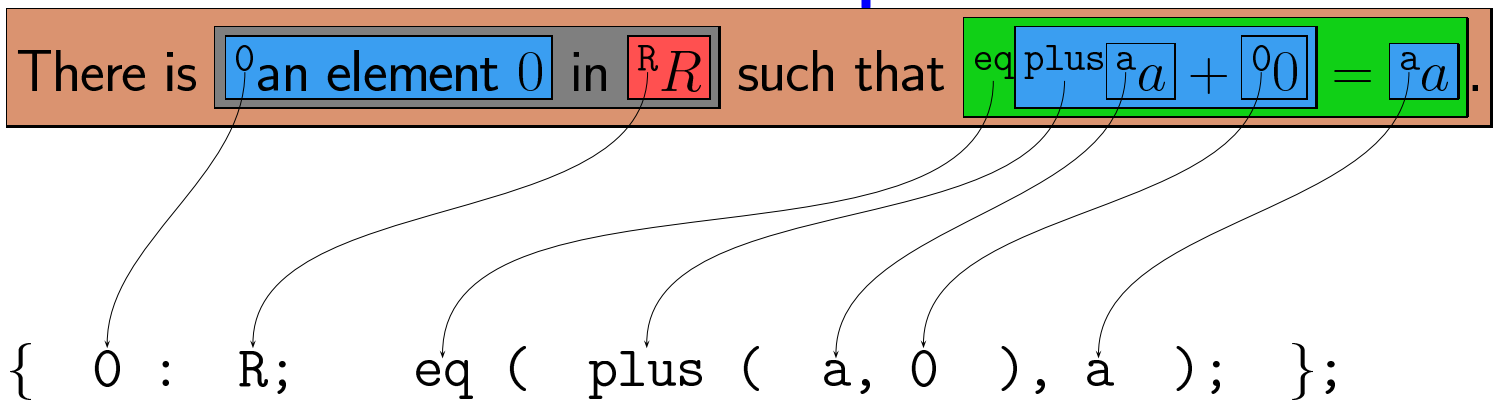
- The type checker for CGa works well and gives some useful error messages. Error messages should be improved.

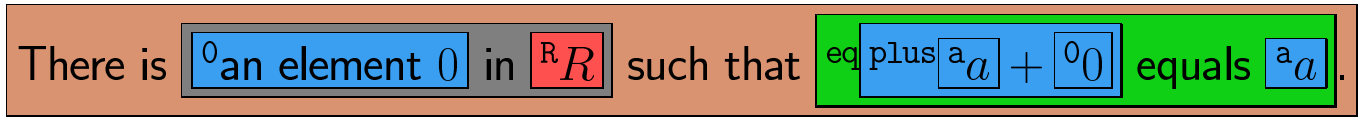

Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

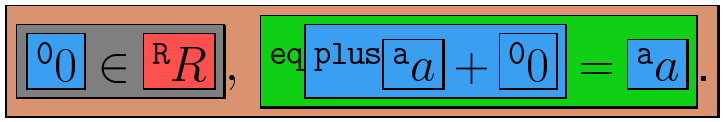
What is TSa? Lamar's PhD thesis

- TSa builds the bridge between a CML text and its grammatical interpretation and adjoins to each CGa expression a string of words and/or symbols which aims to act as its CML representation.
- TSa plays the role of a user interface
- TSa can flexibly represent natural language mathematics.
- The author wraps the natural language text with boxes representing the grammatical categories (as we saw before).
- The author can also give interpretations to the parts of the text.

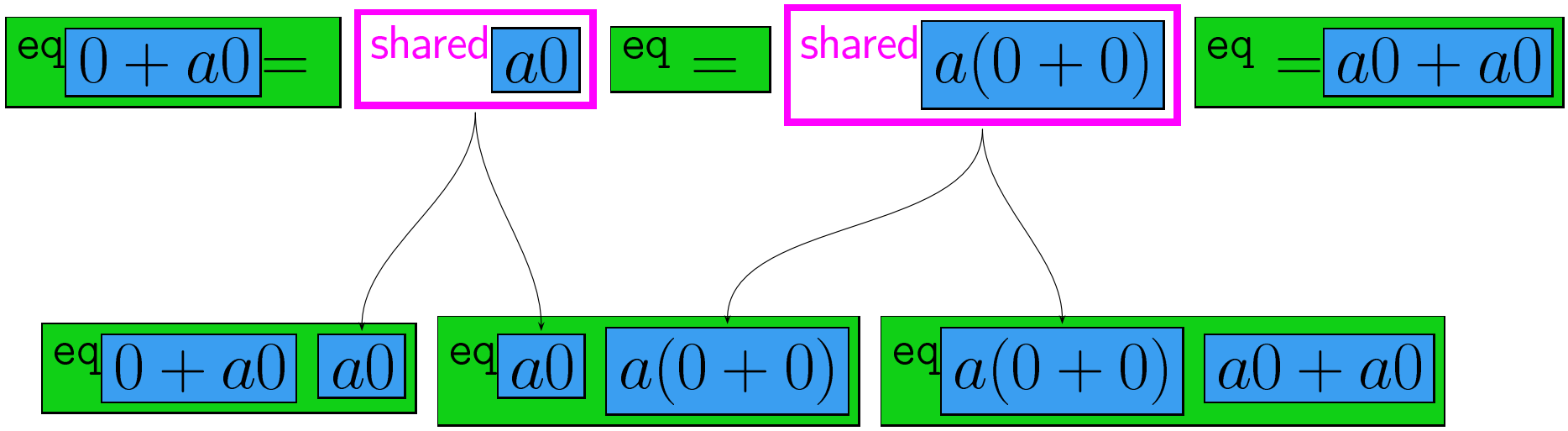
Interpretations







Rewrite rules enable natural language representation

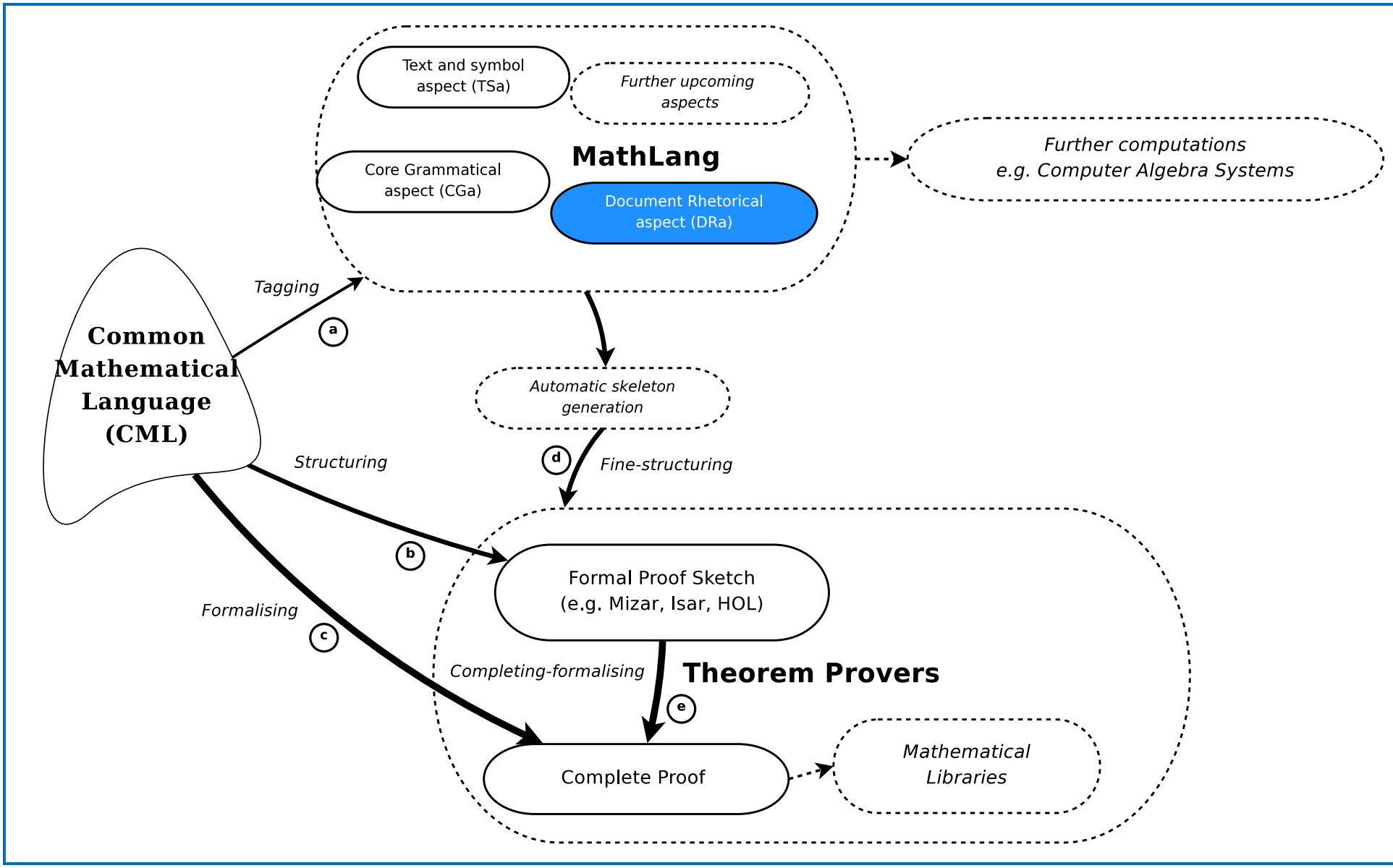


How complete is TSa?

- TSa provides useful interface facilities but it is still under development.

- So far, only simple rewrite (souring) rules are used and they are not comprehensive. E.g., unable to cope with things like $\overbrace{x = \ldots = x}^{n \text{ times}}$.

- The TSa theory and metatheory need development.


Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

What is DRa? Retel's PhD thesis

- DRa Document Rhetorical structure aspect.
- **Structural components of a document** like *chapter, section, subsection, etc.*
- **Mathematical components of a document** like *theorem, corollary, definition, proof, etc.*
- **Relations** between above components.
- These enhance readability, and ease the navigation of a document.
- Also, these help to go into more formal versions of the document.

Relations

Description
Instances of the StructuralRhetoricalRole *class:* preamble, part, chapter, section, paragraph, *etc.*
Instances of the MathematicalRhetoricalRole *class:* lemma, corollary, theorem, conjecture, definition, axiom, claim, proposition, assertion, proof, exercise, example, problem, solution, *etc.*

Relation
Types of relations: relatesTo, uses, justifies, subpartOf, inconsistentWith, exemplifies

What does the mathematician do?

- The mathematician wraps into boxes and uniquely names chunks of text
- The mathematician assigns to each box the structural and/or mathematical rhetorical roles
- The mathematician indicates the relations between wrapped chunks of texts

Lemma 1. For $m, n \in \mathbb{N}$ one has: $m^2 = 2n^2 \implies m = n = 0$.
Define on $\mathbb{N}$ the predicate:

$$P(m) \iff \exists n.m^2 = 2n^2 \ \& \ m > 0.$$

Claim. $P(m) \implies \exists m' < m.P(m')$. Indeed suppose $m^2 = 2n^2$ and $m > 0$. It follows that m^2 is even, but then m must be even, as odds square to odds. So $m = 2k$ and we have

$$2n^2 = m^2 = 4k^2 \implies n^2 = 2k^2$$

Since $m > 0$, if follows that $m^2 > 0, n^2 > 0$ and $n > 0$. Therefore $P(n)$. Moreover, $m^2 = n^2 + n^2 > n^2$, so $m^2 > n^2$ and hence $m > n$. So we can take $m' = n$.

By the claim $\forall m \in \mathbb{N}.\neg P(m)$, since there are no infinite descending sequences of natural numbers.

Now suppose $m^2 = 2n^2$ with $m \neq 0$. Then $m > 0$ and hence $P(m)$. Contradiction. Therefore $m = 0$. But then also $n = 0$.

Corollary 1. $\sqrt{2} \notin \mathbb{Q}$.
Suppose $\sqrt{2} \in \mathbb{Q}$, i.e. $\sqrt{2} = p/q$ with $p \in \mathbb{Z}, q \in \mathbb{Z} - \{0\}$. Then $\sqrt{2} = m/n$ with $m = |p|, n = |q| \neq 0$. It follows that $m^2 = 2n^2$. But then $n = 0$ by the lemma. Contradiction shows that $\sqrt{2} \notin \mathbb{Q}$.

Lemma 1.

For $m, n \in \mathbb{N}$ one has: $m^2 = 2n^2 \Longrightarrow m = n = 0$

Proof.

Define on $\mathbb{N}$ the predicate:

$$P(m) \iff \exists n. m^2 = 2n^2 \ \& \ m > 0.$$

Claim. $P(m) \Longrightarrow \exists m' < m. P(m')$.

Indeed suppose $m^2 = 2n^2$ and $m > 0$. It follows that m^2 is even, but then m must be even, as odds square to odds. So $m = 2k$ and we have $2n^2 = m^2 = 4k^2 \Longrightarrow n^2 = 2k^2$ Since $m > 0$, if follows that $m^2 > 0$, $n^2 > 0$ and $n > 0$. Therefore $P(n)$. Moreover, $m^2 = n^2 + n^2 > n^2$, so $m^2 > n^2$ and hence $m > n$. So we can take $m' = n$.

By the claim $\forall m \in \mathbb{N}. \neg P(m)$, since there are no infinite descending sequences of natural numbers.

Now suppose $m^2 = 2n^2$

with $m \neq 0$. Then $m > 0$ and hence $P(m)$. Contradiction.

Therefore $m = 0$. But then also $n = 0$.

□

Corollary 1. $\sqrt{2} \notin \mathbb{Q}$

Proof. Suppose $\sqrt{2} \in \mathbb{Q}$, i.e. $\sqrt{2} = p/q$ with $p \in \mathbb{Z}, q \in \mathbb{Z} - \{0\}$. Then $\sqrt{2} = m/n$ with $m = |p|, n = |q| \neq 0$. It follows that $m^2 = 2n^2$. But then $n = 0$ by the lemma. Contradiction shows that $\sqrt{2} \notin \mathbb{Q}$. □

$(A, \text{hasMathematicalRhetoricalRole}, \textit{lemma})$
$(E, \text{hasMathematicalRhetoricalRole}, \textit{definition})$
$(F, \text{hasMathematicalRhetoricalRole}, \textit{claim})$
$(G, \text{hasMathematicalRhetoricalRole}, \textit{proof})$
$(B, \text{hasMathematicalRhetoricalRole}, \textit{proof})$
$(H, \text{hasOtherMathematicalRhetoricalRole}, \textit{case})$
$(I, \text{hasOtherMathematicalRhetoricalRole}, \textit{case})$
$(C, \text{hasMathematicalRhetoricalRole}, \textit{corollary})$
$(D, \text{hasMathematicalRhetoricalRole}, \textit{proof})$

$(B, \text{justifies}, A)$
$(D, \text{justifies}, C)$
(D, uses, A)
(G, uses, E)
(F, uses, E)
(H, uses, E)
$(H, \text{subpartOf}, B)$
$(H, \text{subpartOf}, I)$

Lemma 1.

For $m, n \in \mathbb{N}$ one has: $m^2 = 2n^2 \Longrightarrow m = n = 0$ **A**

Proof.

Define on $\mathbb{N}$ the predicate:

$$P(m) \iff \exists n.m^2 = 2n^2 \ \&\ m > 0.$$ **E**

justifies

uses

Claim. $P(m) \Longrightarrow \exists m' < m.P(m')$. **F**

uses

Indeed suppose $m^2 = 2n^2$ and $m > 0$. It follows that m^2 is even, but then m must be even, as odds square to odds. So $m = 2k$ and we have $2n^2 = m^2 = 4k^2 \Longrightarrow n^2 = 2k^2$ Since $m > 0$, it follows that $m^2 > 0$, $n^2 > 0$ and $n > 0$. Therefore $P(n)$. Moreover, $m^2 = n^2 + n^2 > n^2$, so $m^2 > n^2$ and hence $m > n$. So we can take $m' = n$. **G**

justifies

uses

uses

B

By the claim $\forall m \in \mathbb{N}.\neg P(m)$, since there are no infinite descending sequences of natural numbers.

subpartOf

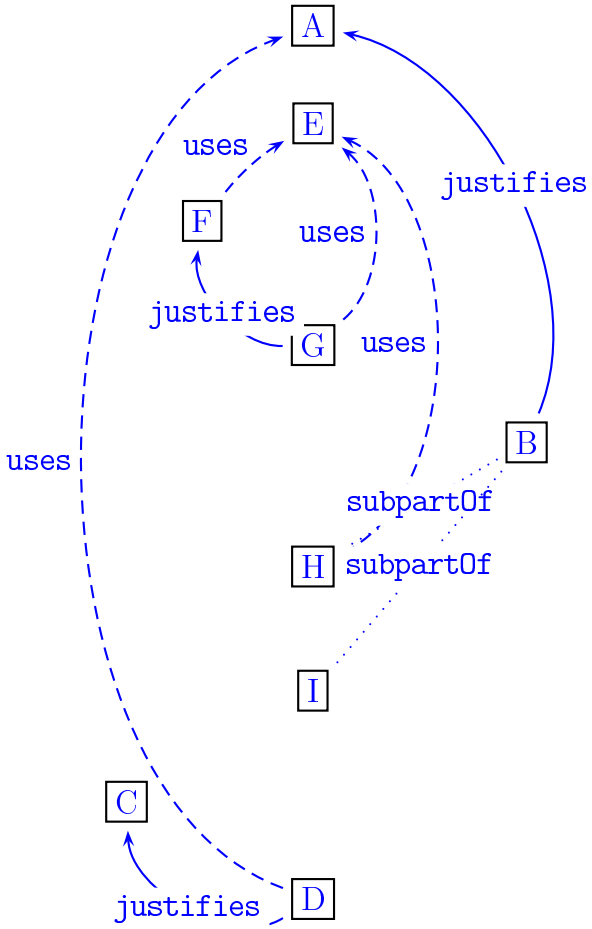
subpartOf

Now suppose $m^2 = 2n^2$

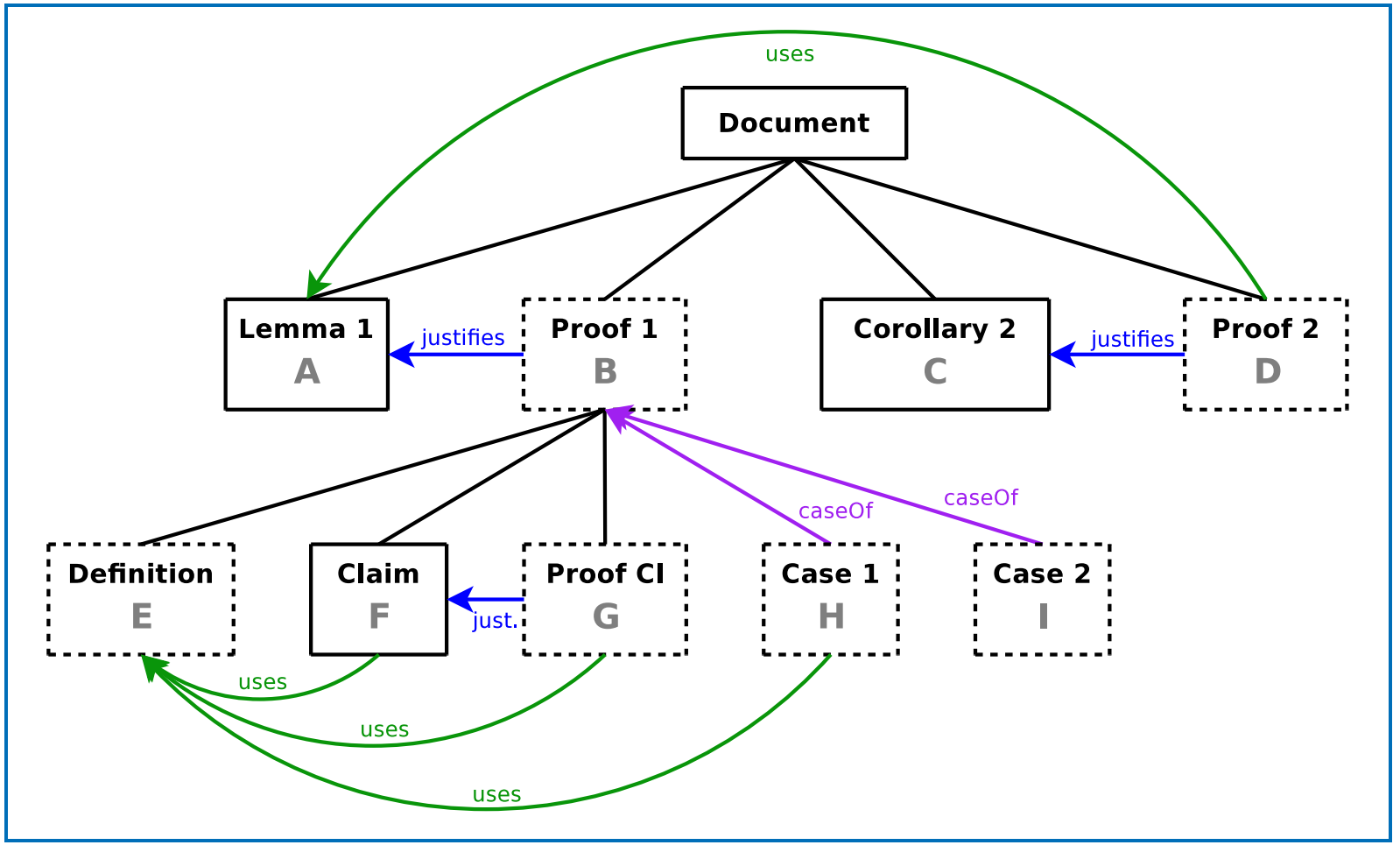
with $m \neq 0$. Then $m > 0$ and hence $P(m)$. Contradiction. **H**

The automatically generated dependency Graph

Dependency Graph (DG)



An alternative view of the DRa

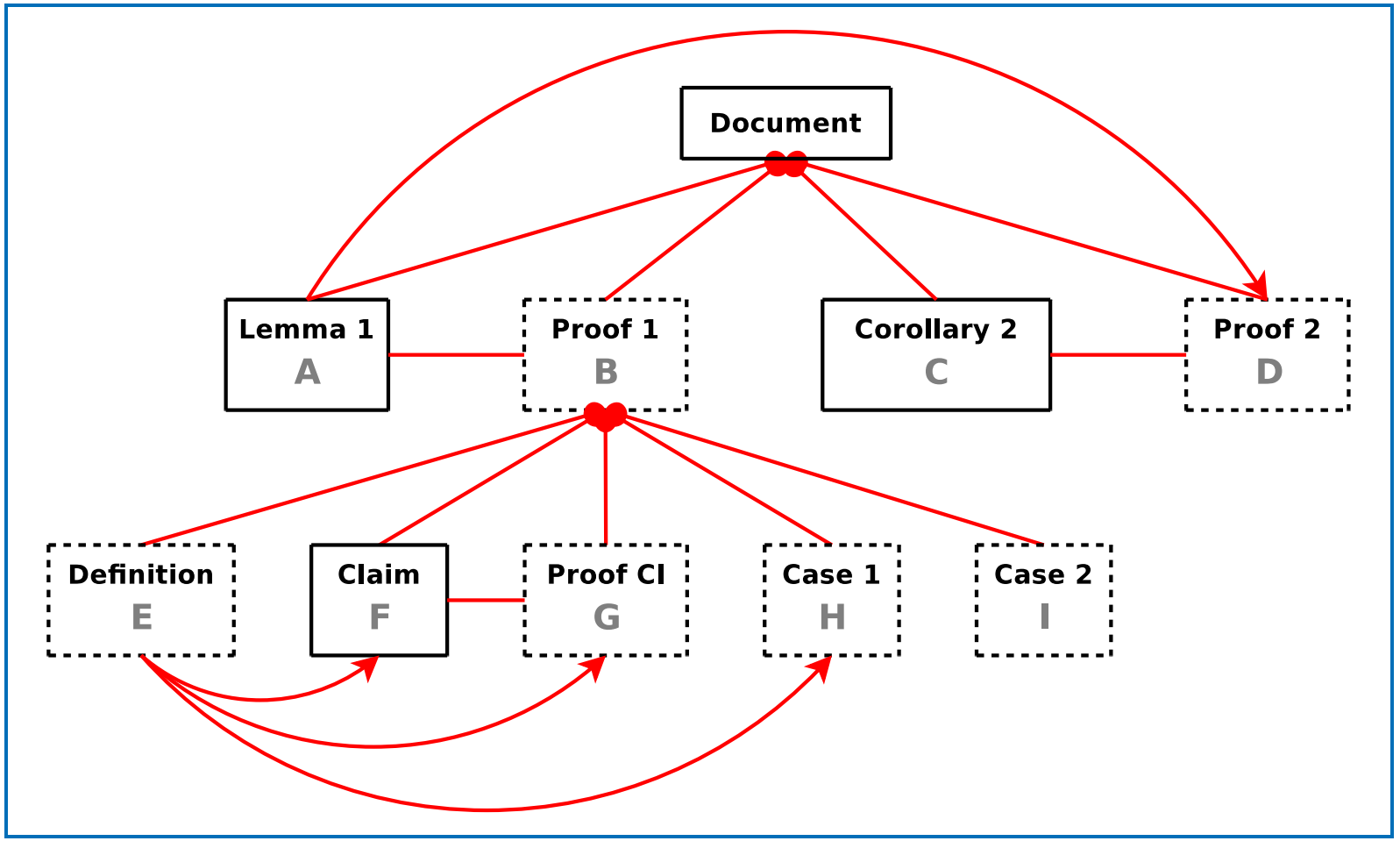


The Graph of Textual Order: GoTO
Zengler's thesis

- To be able to examine the proper structure of a DRa tree we introduce the concept of textual order between two nodes in the tree.
- Using textual orders, we can transform the dependency graph into a GoTO by transforming each edge of the DG.
- So far there are two reasons why the GoTO is produced:
  1. Automatic Checking of the GoTO can reveal errors in the document (e.g. loops in the structure of the document).
  2. The GoTO is used to automatically produce a proof skeleton for a certain prover.
- We automatically transform a DG into GoTO and automatically check the GoTO for errors in the document:

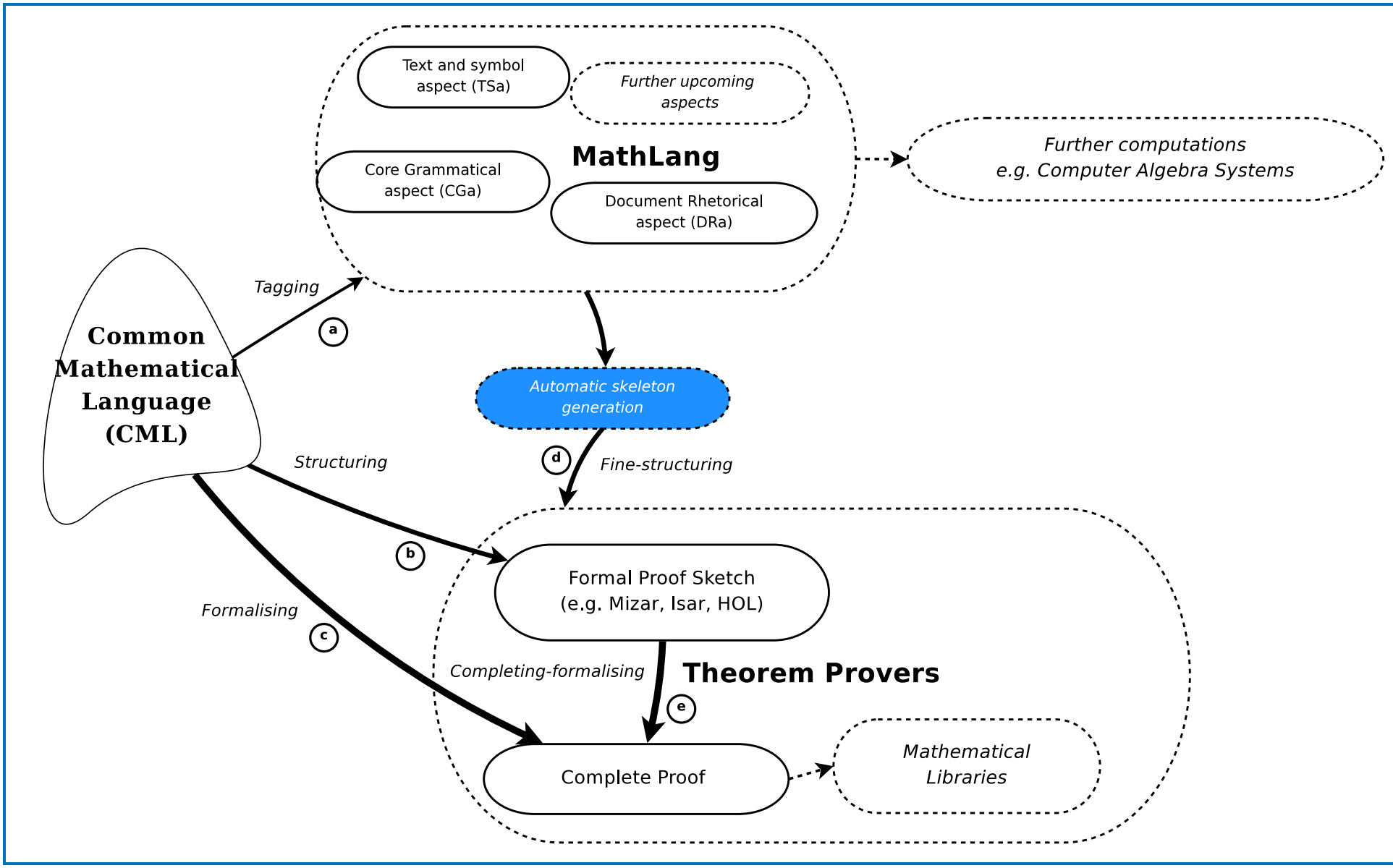
1. Loops in the GoTO (error)
2. Proof of an unproved node (error)
3. More than one proof for a proved node (warning)
4. Missing proof for a proved node (warning)

Graph of Textual Order for the DRa tree example



How complete is DRa?

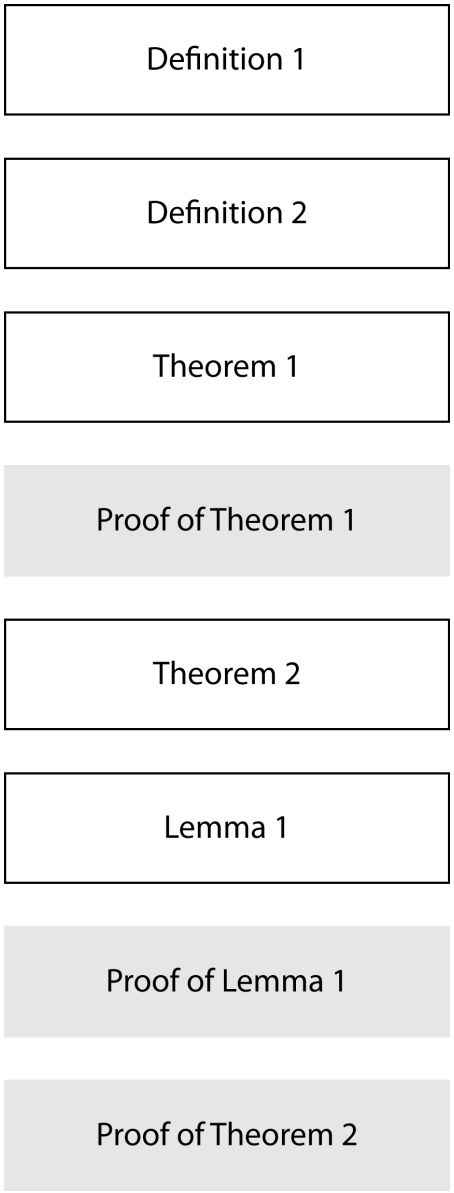
- The dependency graph can be used to check whether the logical reasoning of the text is coherent and consistent (e.g., no loops in the reasoning).
- However, both the DRa language and its implementation need more experience driven tests on natural language texts.
- Also, the DRa aspect still needs a number of implementation improvements (the automation of the analysis of the text based on its DRa features).
- Extend TSa to also cover DRa (in addition to CGa).
- Extend DRa depending on further experience driven translations.
- Establish the soundness and completeness of DRa for mathematical texts.


Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

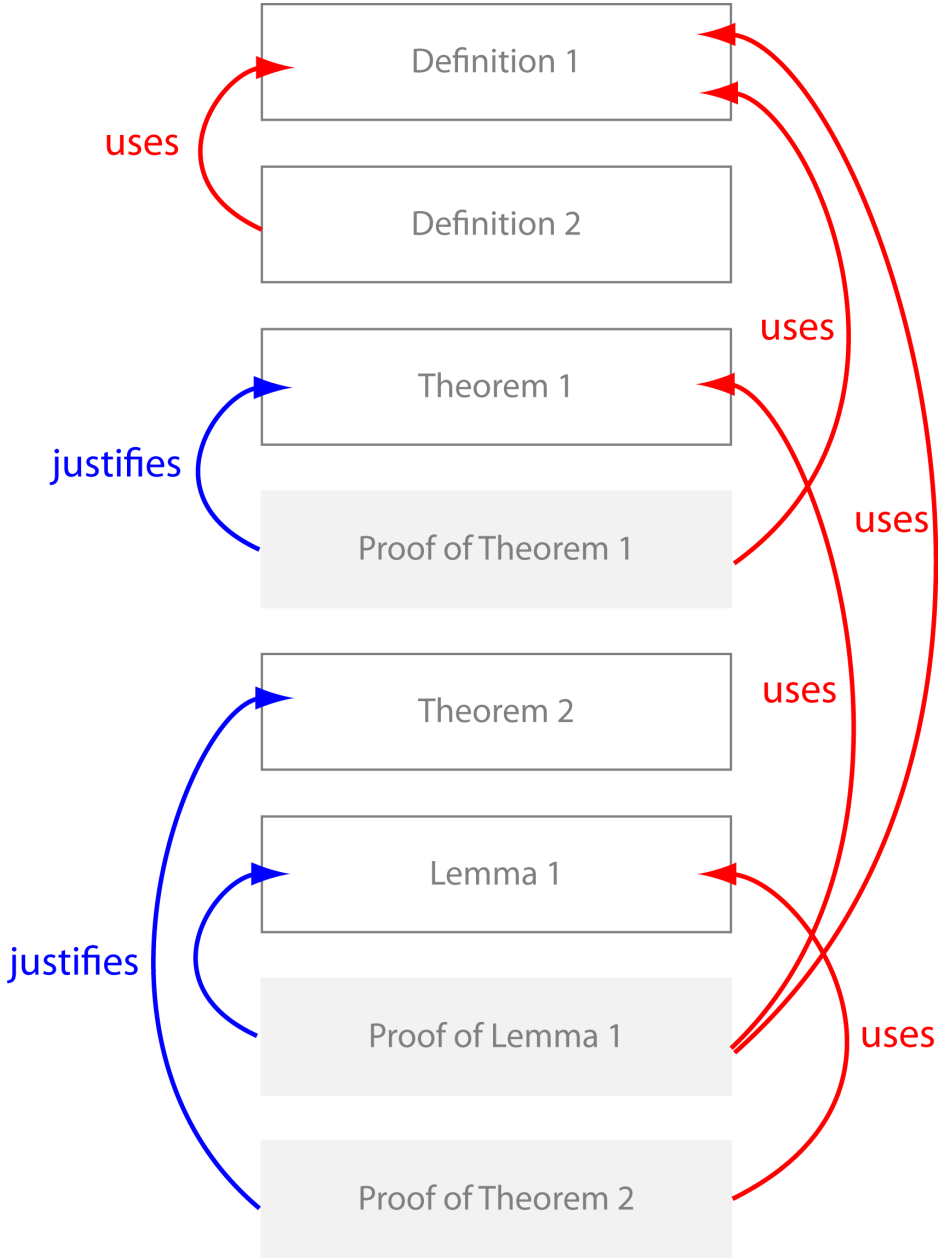
Different provers have

- different syntax
- different requirements to the structure of the text
  e.g.
  - no nested theorems/lemmas
  - only backward references
  - ...
- Aim: Skeleton should be as close as possible to the mathematician's text but with re-arrangements when necessary

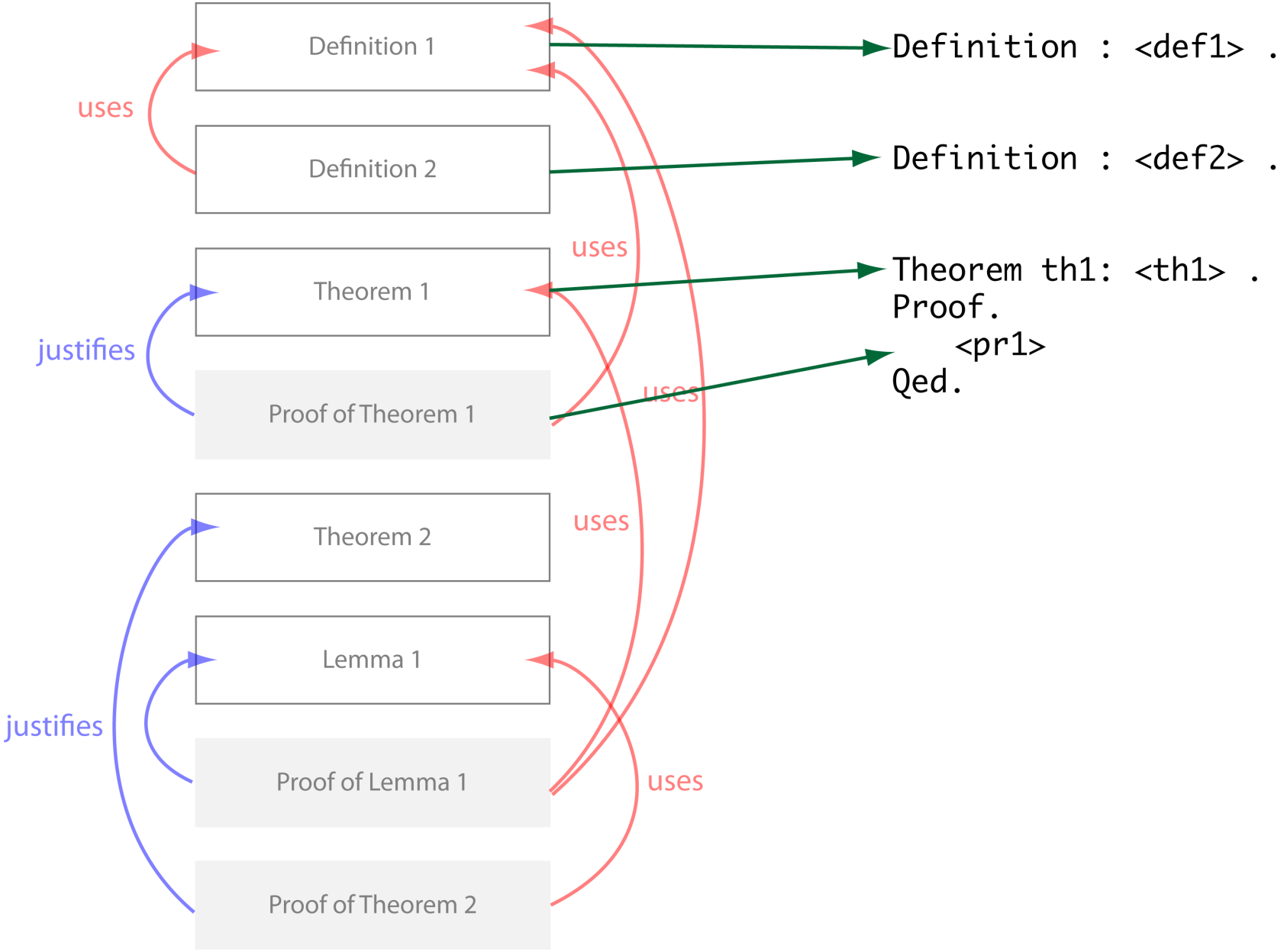
Example of nested theorems/lemmas (Moller, 03, Chapter III,2)



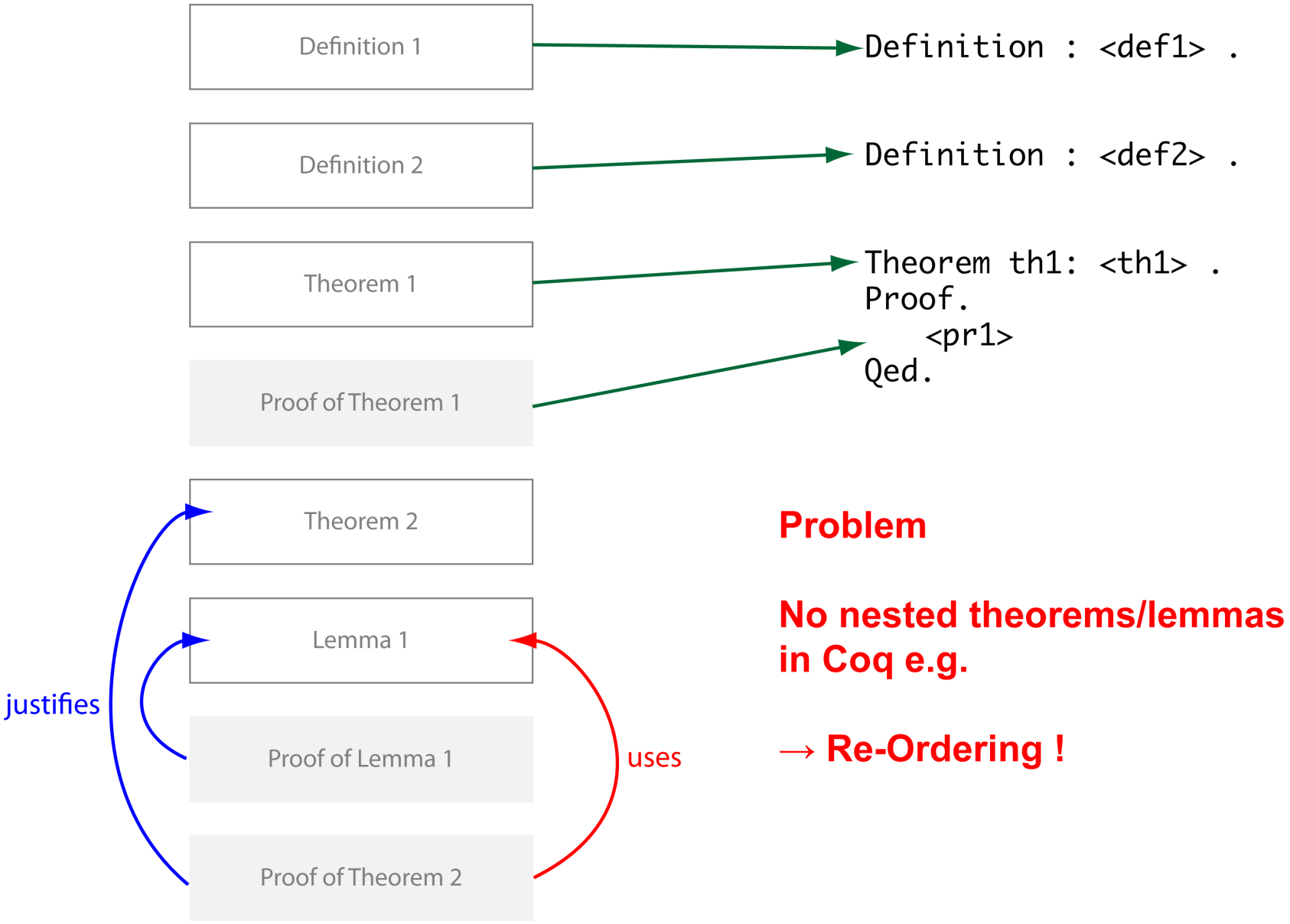
The automatic generation of a proof skeleton



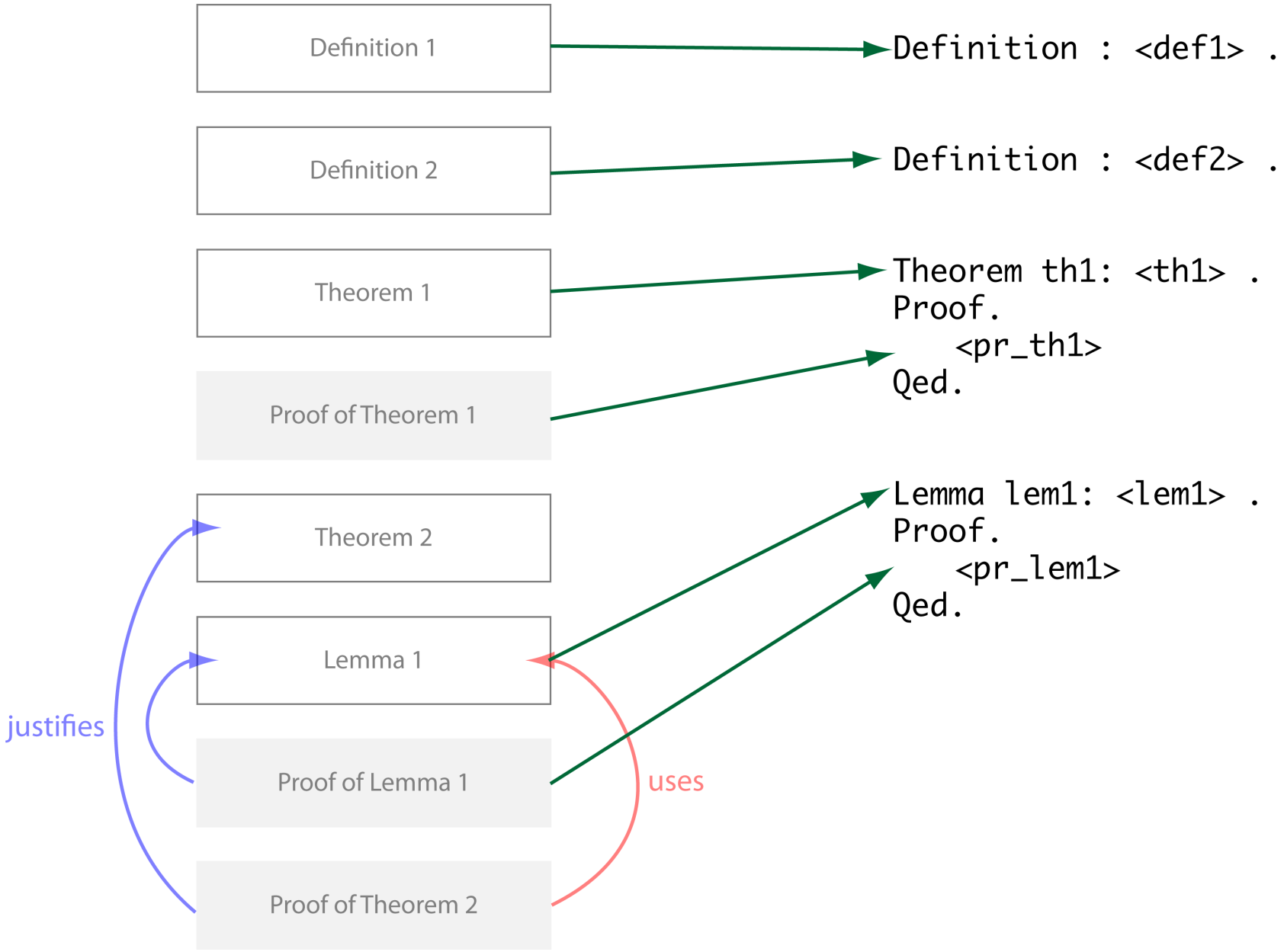
The DG for the example



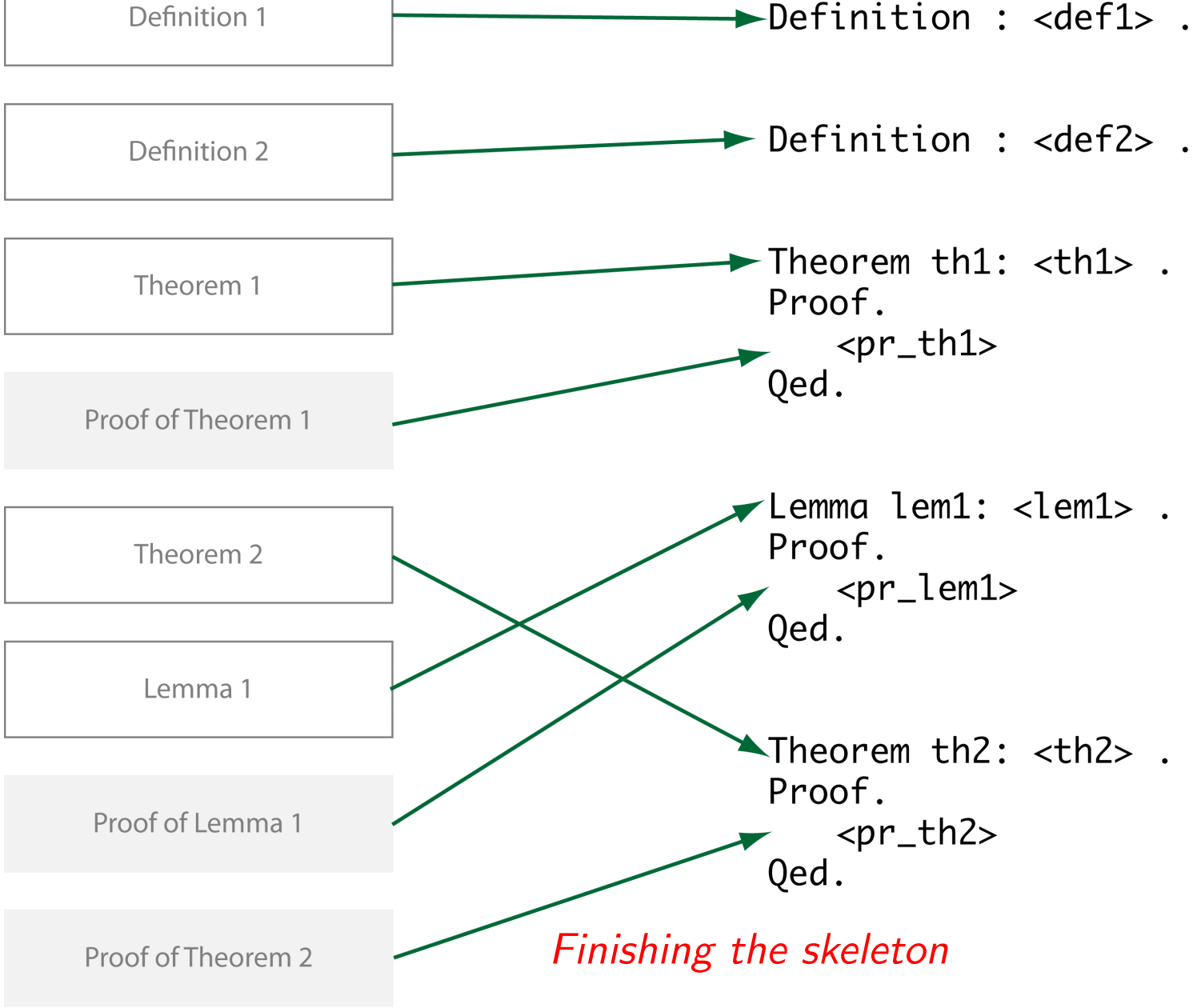
Straight-forward translation of the first part

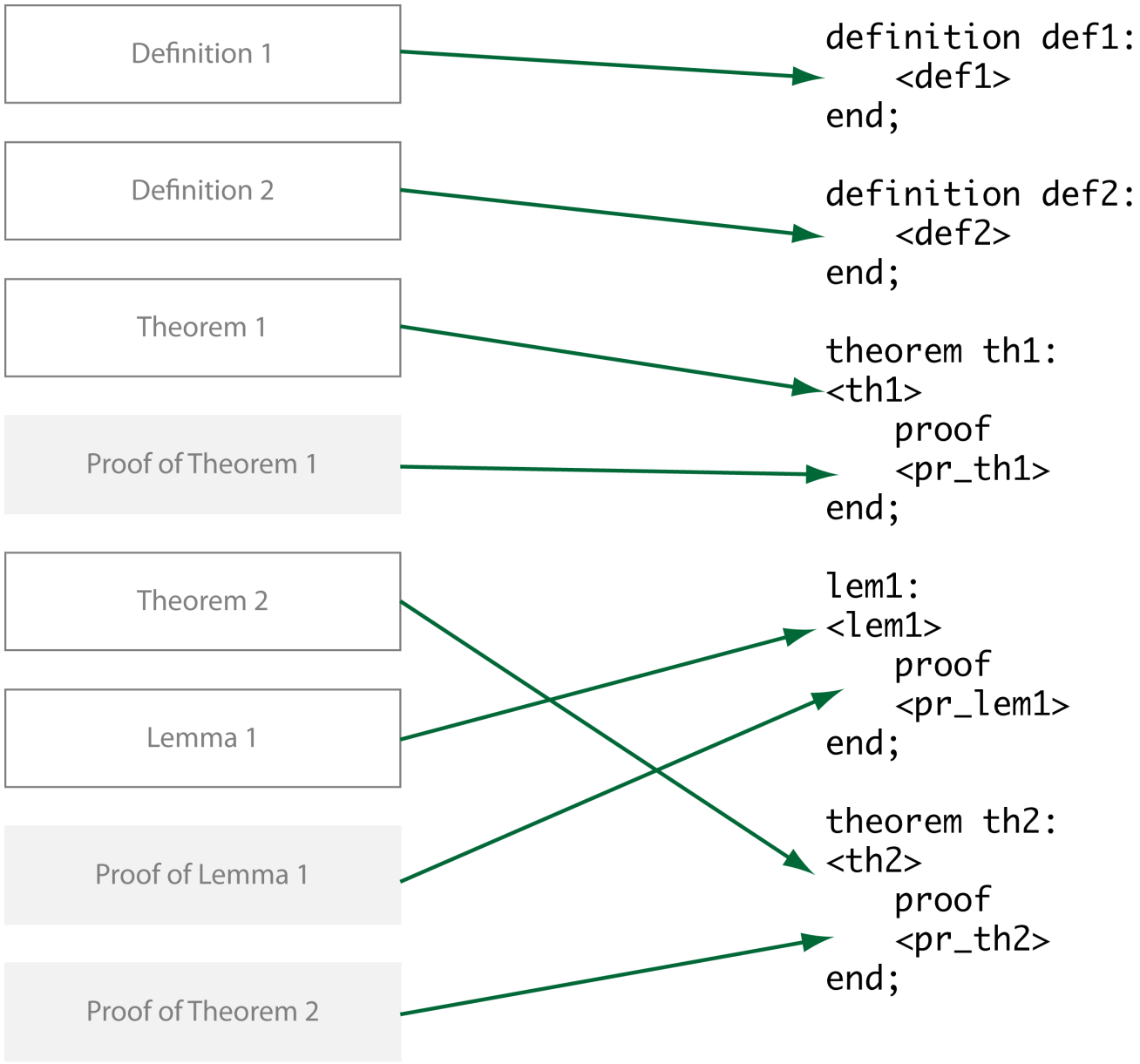


Problem: nested theorems



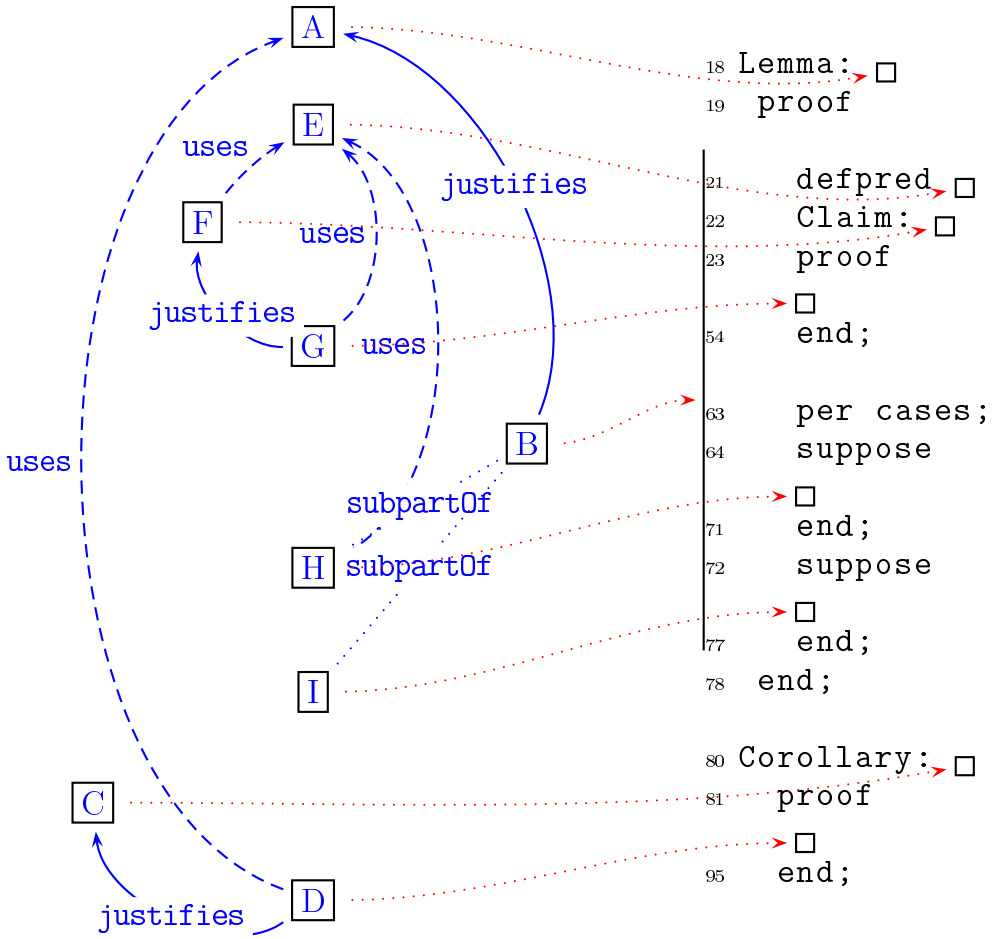
Solution: Re-ordering


Definition 1
Definition 2
Theorem 1
Proof of Theorem 1
Theorem 2
Lemma 1
Proof of Lemma 1
Proof of Theorem 2
Definition : <def1> .
Definition : <def2> .
Theorem th1: <th1> .
Proof.
<pr_th1>
Qed.
Lemma lem1: <lem1> .
Proof.
<pr_lem1>
Qed.
Theorem th2: <th2> .
Proof.
<pr_th2>
Qed.
Finishing the skeleton


Definition 1
Definition 2
Theorem 1
Proof of Theorem 1
Theorem 2
Lemma 1
Proof of Lemma 1
Proof of Theorem 2
definition def1:
<def1>
end;
definition def2:
<def2>
end;
theorem th1:
<th1>
proof
<pr_th1>
end;
lem1:
<lem1>
proof
<pr_lem1>
end;
theorem th2:
<th2>
proof
<pr_th2>
end;

Skeleton for Mizar

DRa annotation into Mizar skeleton for Barendregt's example (Retel's PhD thesis)



The remaining very rough path into Mizar

- We have not built the remaining aspects all the way into Mizar, but we have a rough path.
- Recall that GoTO gives a Mizar skeleton of the text.
- Next, the CGa encoding of the text is used to build relevant parts of the Mizar FPS (Wiedijk 2003) of the text (e.g., the CGa **preamble** could be used to find counterparts in Mizar MML and to build parts of the *Environment* in Mizar).
- At this stage, a Mizar expert would be able to complete the Mizar FPS version of the text.
- Now, the Mizar experts can complete the formalisation by filling all the gaps in the reasoning (i.e., filling the holes in sentences labelled with the error *4 by the Mizar system.)

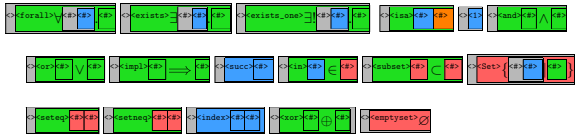
The Mizar FPS version of Barendregt's example

```
20 Lemma:  for m,n being Nat holds
21                m^2 = 2*n^2 implies m = 0 & n = 0
22  proof
23    let m,n being Nat;
24    defpred P[Nat] means
25             ex n being Nat st $1^2 = 2*n^2 & $1 > 0;
26    Claim: for m being Nat holds
27           P[m] implies ex m' being Nat st m' < m & P[m']
28    proof
29      let m being Nat;
30      assume P[m];
31      then consider n being Nat such that
32      m^2 = 2*n^2 & m > 0;
33      m^2 is even ;
34 ::>             *4
35      m is even;
36 ::>           *4
37      consider k being Nat such that m = 2*k;
38 ::>                                        *4
39      2*n^2 = m^2
40 ::>             *4
41           .= 4*k^2;
42 ::>               *4
43      then n^2 = 2*k^2;
44      m > 0 implies m^2 > 0 & n^2 > 0 & n > 0;
45 ::>                                       *4,4,4
46      then P[n];
47 ::>           *4,4
48      m^2 = n^2 + n^2;
49 ::>                *4
50      n^2 + n^2 >  n^2;
51 ::>                  *4
52      then m^2 > n^2;
53 ::>                *4
54      then m > n;
55 ::>             *4
56      take m' = n;
57      thus thesis;
58 ::>             *4,4
59    end;
```

```
67    A2: for k being Nat holds not P[k]
68    proof
69      not ex q being Seq_of_Nat
70          st q is infinite decreasing by Claim;
71 ::>                                          *4
72      hence thesis;
73 ::>              *4
74    end;
75    assume A0: m^2 = 2*n^2;
76    per cases by A0;
77    suppose B1: m <> 0;
78      then m > 0;
79 ::>            *4
80      then P[m] by B1;
81 ::>              *4
82      then contradiction by A2;
83      hence thesis;
84    end;
85    suppose S1: m = 0;
86      then n = 0;
87 ::>            *4
88      thus thesis by S1;
89 ::>                *4
90    end;
91  end;
92
93 Corollary:  sqrt 2 is irrational
94   proof
95     assume sqrt 2 is rational;
96     then ex p,q being Integer st
97     q <> 0 & sqrt 2 = p/q;
98 ::>                        *4
99     then consider m,n being Integer such that
100    A0: sqrt 2 = m/n & m = abs m & n = abs n & n <> 0;
101 ::>                                                *4
102    m^2 = 2*n^2;
103 ::>           *4
104    n = 0 by Lemma;
105 ::>         *4
106    hence contradiction;
107 ::>                    *4
108  end;
109
110 ::> 4: This inference is not accepted
```

Chapter 1

Natural Numbers



1.1 Axioms

We assume the following to be given:

A set (i.e. totality) of objects called natural numbers, possessing the properties - called axioms- to be listed below.

Before formulating the axioms we make some remarks about the symbols $=$ and $\neq$ which be used.

Unless otherwise specified, small italic letters will stand for natural numbers throughout this book.

If x is given and y is given, then either x and y are the same number; this may be written

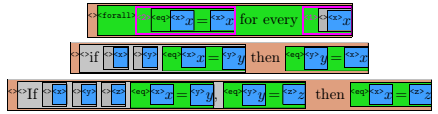
$$x = y$$

($=$ to be read “equals"); or x and y are not the same number; this may be written

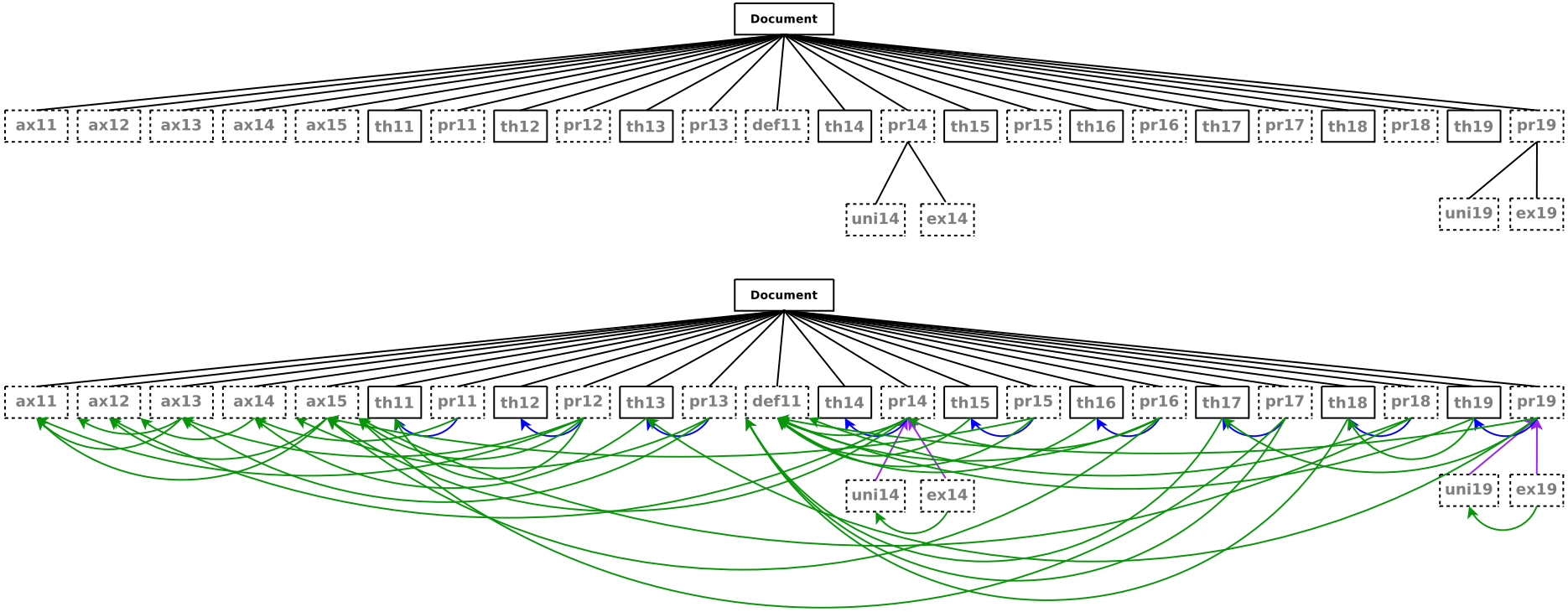
$$1x \neq y$$

($\neq$ to be read “is not equal to").

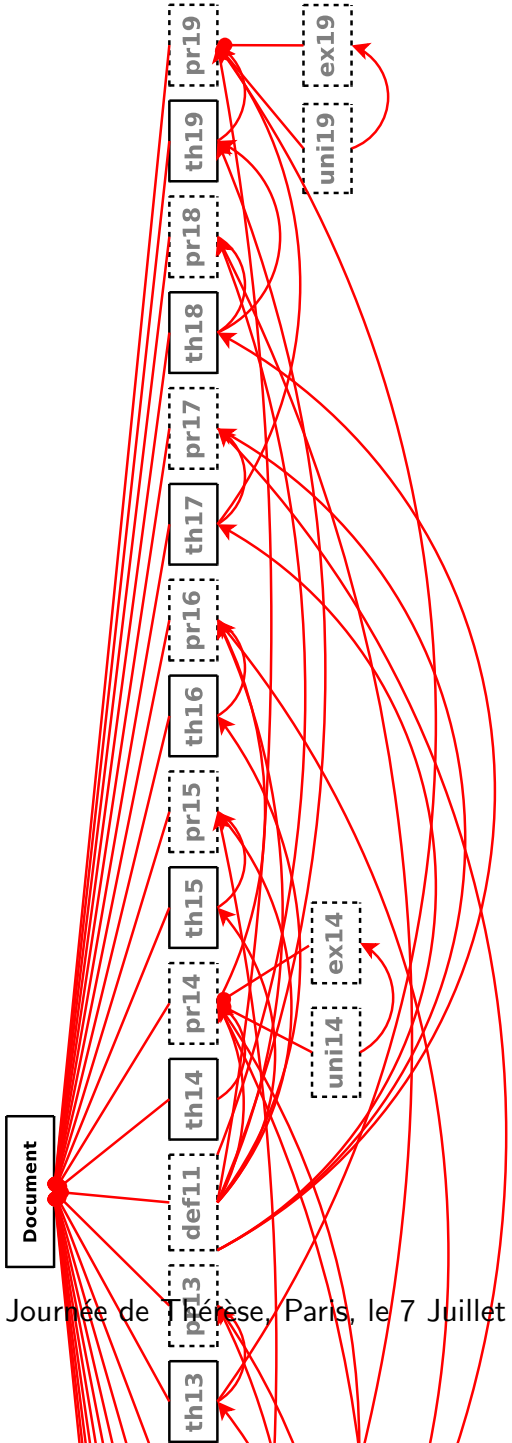
Accordingly, the following are true on purely logical grounds:



The DRa tree and the DG of sections 1 and 2 of chapter 1 of Landau's book



The GoTO of sections 1 and 2 of chapter 1 of Landau's book


Document
def11
th13
pr13
th14
pr14
th15
pr15
th16
pr16
th17
pr17
th18
pr18
th19
pr19
ex14
uni14
ex19
uni19

Extending proof skeletons with CGa hints

- Translations into Coq for large parts of the chapter can be automated. E.g.,by just viewing the interpretation of the annotations of axiom 3 we get:

  forall x (neq (succ(x), 1)) (a)

  The automatically generated Coq proof skeleton for this axiom is:

  Axiom ax13 : <ax13> . (b)

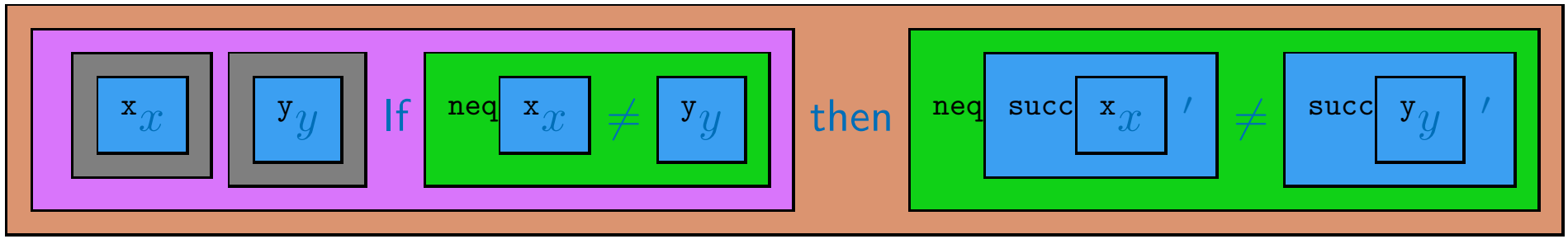
  Now, we simply replace the <ax13> placeholder of (b) with the literal translation of the interpretations in (a) to get the valid Coq axiom:

  Axiom ax13 : forall x:nats, neq (succ x) I .

Similarly for the theorems of chapter 1 of Landau's book, the work needed to get the full formalisation is straightforward: E.g. Theorem 1 is written by Landau as:

$$\text{If } x \neq y \text{ then } x' \neq y'$$

Its annotation in MathLang CGa is:



The CGa annotation of the context can also be seen as the premise of an implication. So the upper statement can be translated to:

$$\text{decl(x), decl(y) : neq x y} \rightarrow \text{neq (succ x) (succ y)}$$

And when we compare this line with its Coq translation we see again, it is just a literal transcription of the interpretation parts of CGa and therefore could be easily performed by an algorithm.

$$\text{Theorem th11 (x y:nats) : neq x y} \longrightarrow \text{neq (succ x) (succ y) .}$$

From the 36 theorems of the chapter 28 could be translated literally into their corresponding Coq theorems.

Some points to consider

- We do not at all assume/prefer one type/logical theory instead of another.
- The formalisation of a language of mathematics should separate the questions:
  - *which type/logical theory is necessary for which part of mathematics*
  - *which language should mathematics be written in*.
- Mathematicians don't usually know or work with type/logical theories.
- Mathematicians usually *do* mathematics (manipulations, calculations, etc), but are not interested in general in reasoning *about* mathematics.
- The steps used for computerising books of mathematics written in English, as we are doing, can also be followed for books written in Arabic, French, German, or any other natural language.

- MathLang aims to support non-fully-formalized mathematics practiced by the ordinary mathematician as well as work toward full formalization.
- MathLang aims to handle mathematics as expressed in natural language as well as symbolic formulas.
- MathLang aims to do some amount of type checking even for non-fully-formalized mathematics. This corresponds roughly to grammatical conditions.
- MathLang aims for a formal representation of CML texts that closely corresponds to the CML conceived by the ordinary mathematician.
- MathLang aims to support automated processing of mathematical knowledge.
- MathLang aims to be independent of any foundation of mathematics.
- MathLang allows anyone to be involved, whether a mathematician, a computer engineer, a computer scientist, a linguist, a logician, etc.

Thérèse

- I discussed how influential Thérèse has been in developing new ideas (explicit substitutions, rewriting, generalised proofs of confluence, reducing HOU to first order unification, working modulo classes rather than elements of these classes.

- I did not have much time to discuss her influential system of Focal, but this system is "live and kicking" and a lot of interesting work has and still is being done in it.

- With the increasing demand for cerifiably secure systems, the work of Thérèse and her team on establishing security properties using Focal can teach many lessons here.

- I am proud to have been influenced by the work of Thérèse.

Bibliography

Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. In *Conf. Rec. 22nd Ann. ACM Symp. Princ. of Prog. Langs.*, pages 233–246, 1995.

Mauricio Ayala-Rincón and Fairouz Kamareddine. Unification via the lambda s_e-style of explicit substitutions. *Logic Journal of the IGPL*, 9(4), 2001.

H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, revised edition, 1984.

Z.E.A. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, 1996.

Roel Bloo, Fairouz Kamareddine, and Rob Nederpelt. The Barendregt cube with definitions and generalised reduction. *Inform. & Comput.*, 126(2):123–143, May 1996.

N.G. de Bruijn. The mathematical language AUTOMATH, its usage and some of its extensions. In M. Laudet, D. Lacombe, and M. Schuetzenberger, editors, *Symposium on Automatic Demonstration*, pages 29–61, IRIA, Versailles, 1968. Springer Verlag, Berlin, 1970. Lecture Notes in Mathematics **125**; also in [**?**], pages 73–100.

A. Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *J. ACM*, 43(2):362–397, 1996.

Philippe de Groote. The conservation theorem revisited. In *Proc. Int'l Conf. Typed Lambda Calculi and Applications*, pages 163–178. Springer, 1993.

Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher order unification via explicit substitutions. *Inf. Comput.*, 157(1-2):183–235, 2000.

Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. *J. Autom. Reasoning*, 31(1):33–72, 2003.

J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

Thérèse Hardin. Confluence results for the pure strong categorical logic ccl: lambda-calculi as subsystems of ccl. *Theor. Comput. Sci.*, 65(3):291–342, 1989.

R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Proceedings Second Symposium on Logic in Computer Science*, pages 194–204, Washington D.C., 1987. IEEE.

J.R. Hindley and J.P. Seldin. *Introduction to Combinators and λ-calculus*, volume 1 of *London Mathematical Society Student Texts*. Cambridge University Press, 1986.

F. Kamareddine and R. Bloo. De Bruijn's syntax and reductional behaviour of lambda terms. *Submitted*, 2002.

F. Kamareddine, R. Bloo, and R. Nederpelt. On Π-conversion in the λ-cube and the combination with abbreviations. *Ann. Pure Appl. Logic*, 97(1–3):27–45, 1999.

F. Kamareddine, L. Laan, and R.P. Nederpelt. Refining the Barendregt cube using parameters. In *Proceedings of the Fifth International Symposium on Functional and Logic Programming, FLOPS 2001*, pages 375–389, 2001a.

F. Kamareddine, T. Laan, and R. P. Nederpelt. Types in logic and mathematics before 1940. *Bulletin of Symbolic Logic*, 8(2):185–245, June 2002.

F. Kamareddine, T. Laan, and R. P. Nederpelt. Revisiting the λ-calculus notion of function. *J. Algebraic & Logic Programming*, 54:65–107, 2003.

Fairouz Kamareddine. Postponement, conservation and preservation of strong normalisation for generalised reduction. *J. Logic Comput.*, 10(5):721–738, 2000.

Fairouz Kamareddine and Rob Nederpelt. On stepwise explicit substitution. *Int'l J. Foundations Comput. Sci.*, 4(3): 197–240, 1993.

Fairouz Kamareddine and Rob Nederpelt. Refining reduction in the λ-calculus. *J. Funct. Programming*, 5(4): 637–651, October 1995.

Fairouz Kamareddine and Rob Nederpelt. A useful λ-notation. *Theoret. Comput. Sci.*, 155(1):85–109, 1996.

Fairouz Kamareddine and Alejandro Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *J. Funct. Programming*, 7(4):395–420, 1997.

Fairouz Kamareddine and Alejandro Ríos. A λ-calculus à la de Bruijn with explicit substitution. In *7th Int'l Symp. Prog. Lang.: Implem., Logics & Programs, PLILP '95*, volume 982 of *Lecture Notes in Computer Science*, pages 45–62. Springer, 1995.

Fairouz Kamareddine, Alejandro Ríos, and J. B. Wells. Calculi of generalised β-reduction and explicit substitutions: The type free and simply typed versions. *J. Funct. Logic Programming*, 1998(5), June 1998.

Fairouz Kamareddine, Roel Bloo, and Rob Nederpelt. De Bruijn's syntax and reductional equivalence of lambda terms. In *Proc. 3rd Int'l Conf. Principles & Practice Declarative Programming*, 5–7 September 2001b. ISBN 1-58113-388-X.

A. J. Kfoury and J. B. Wells. A direct algorithm for type inference in the rank-2 fragment of the second-order λ-calculus. In *Proc. 1994 ACM Conf. LISP Funct. Program.*, pages 196–207, 1994. ISBN 0-89791-643-3.

A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi. In *Proc. 10th Ann. IEEE Symp. Logic in Comput. Sci.*, pages 311–321, 1995. ISBN 0-8186-7050-9. URL `http://www.church-project.org/reports/electronic/Kfo+Wel:LICS-1995.pdf.%gz`.

Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. An analysis of ML typability. *J. ACM*, 41(2):368–398, March 1994.

G. Longo and E. Moggi. Constructive natural deduction and its modest interpretation. Technical Report CMU-CS-88-131, Carnegie Mellono University, Pittsburgh, USA, 1988.

Rob Nederpelt. *Strong Normalization in a Typed Lambda Calculus With Lambda Structured Types*. PhD thesis, Eindhoven, 1973.

R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer, editors. *Selected Papers on Automath*. Studies in Logic and the Foundations of Mathematics **133**. North-Holland, Amsterdam, 1994.

L. Regnier. Une équivalence sur les lambda termes. *Theoretical Computer Science*, 126:281–292, 1994.

Laurent Regnier. *Lambda calcul et réseaux*. PhD thesis, University Paris 7, 1992.

G.R. Renardel de Lavalette. Strictness analysis via abstract interpretation for recursively defined types. *Information and Computation*, 99:154–177, 1991.

J.C. Reynolds. *Towards a theory of type structure*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer, 1974.