



On the Dynamic Increase of Multiplicities in Matrix Proof Methods for Classical Higher-Order Logic

Serge Autexier

`serge@ags.uni-sb.de`

DFKI GmbH & CS Department, Saarland University, Saarbrücken, Germany

Tableaux'05, September 17th, 2005

Koblenz, Germany

Context



- Work on the CORE calculus [[CADE05](#)]
- Needed a uniform mechanism to check admissibility of substitutions
- Used uniform integration of Expansion trees [[Miller83,Pfenning89](#)]/Indexed formula trees [[Wallen90](#)]
- Needed a uniform mechanism to increase multiplicities on the fly without having to restart
⇒ This talk: illustrate this for extensional expansion trees for HOL with equality

Expansion Proofs



- To prove a formula φ
 1. Choose a *multiplicity* for the γ -quantifiers in φ^+ , build an initial *expansion tree*
 2. Then search for an *admissible substitution* such that all *paths* have a *connection*
 - ▶ In higher-order logic quantifiers can be introduced by substitutions
 - ▶ In the presence of equality need to apply *rules to deal with equality*
 3. If 2. fails, go back to 1.

Expansion Proofs



- To prove a formula φ
 1. Choose a *multiplicity* for the γ -quantifiers in φ^+ , build an initial *expansion tree*
 2. Then search for an *admissible substitution* such that all *paths* have a *connection*
 - ▶ In higher-order logic quantifiers can be introduced by substitutions
 - ▶ In the presence of equality need to apply *rules to deal with equality*
 3. If 2. fails, go back to 1.

- Problem:
 - ▶ Having to restart happens more often than not
 - ▶ When restarting any information is lost (substitutions, connections, =-rules)

Expansion Proofs

- To prove a formula φ
 1. Choose a *multiplicity* for the γ -quantifiers in φ^+ , build an initial *expansion tree*
 2. Then search for an *admissible substitution* such that all *paths* have a *connection*
 - ▶ In higher-order logic quantifiers can be introduced by substitutions
 - ▶ In the presence of equality need to apply *rules to deal with equality*
 3. If 2. fails, go back to 1.
- Problem:
 - ▶ Having to restart happens more often than not
 - ▶ When restarting any information is lost (substitutions, connections, =-rules)
- Solution:
 - ▶ Include adjustment of multiplicities into the kernel procedure 2.
 - ▶ Allows to start from any multiplicity without having to restart
 - ▶ Better suited to deal with quantifiers introduced by substitution

Extensional Expansion Trees

Extensional Expansion Trees [Pfenning89] are build from

- Initial expansion trees for the given formula with a chosen multiplicity
- Rules operating on the trees
 - ▶ Substitution (necessary to be actually applied for HOL)
 - ▶ Introduction of Leibniz' definition of equality (for equality handling)
 - ▶ Extensionality rule for equality

Here we use a presentation close to Wallens *indexed formula trees*

- Use polarities and uniform notation $\alpha, \beta, \gamma, \delta$
- Notation for expansion trees (p polarity)

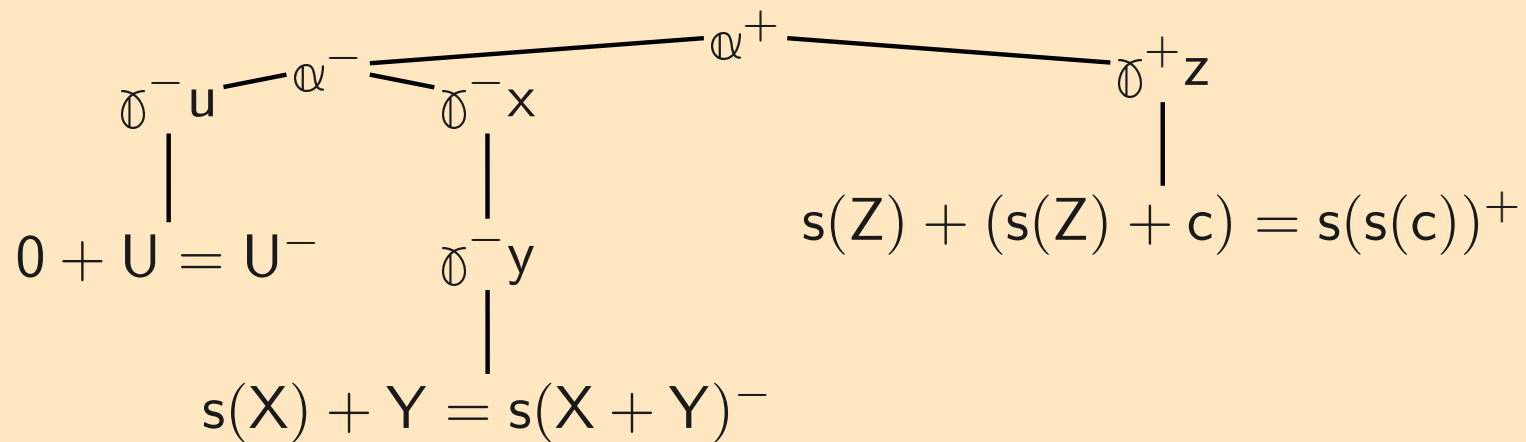
$$Q(t_1, \dots, t_n)^p, \alpha^p(\Delta, \Delta'), \beta^p(\Delta, \Delta'), \delta^{p \times}(\Delta_{x_1}, \dots, \Delta_{x_n}), \delta^{p \times}(\Delta)$$
- Each node has a label (the original formula) and a *deep formula*

Example

- *Formula:*

$$((\forall u . 0 + u = u) \wedge \forall x . \forall y . s(x) + y = s(x + y)) \Rightarrow \exists z . s(z) + (s(z) + c) = s(s(c))$$

- *Initial Extensional Expansion Tree:* Multiplicity 1



- *Deep Formula:*

$$((0 + U = U) \wedge s(X) + Y = s(X + Y))$$

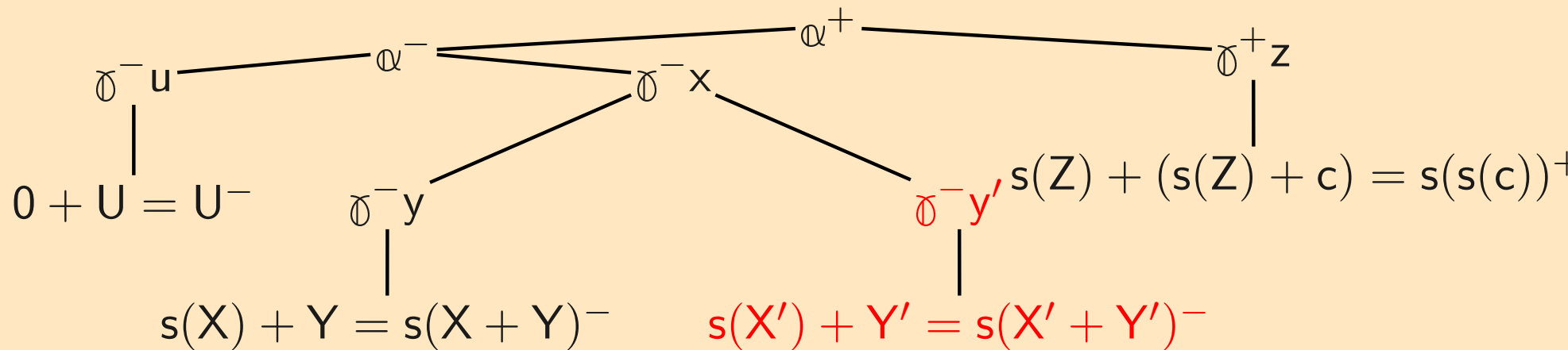
$$\Rightarrow s(Z) + (s(Z) + c) = s(s(c))$$

Example

- *Formula:*

$$((\forall u . 0 + u = u) \wedge \forall x . \forall y . s(x) + y = s(x + y)) \Rightarrow \exists z . s(z) + (s(z) + c) = s(s(c))$$

- *Initial Extensional Expansion Tree:* Multiplicity 2 for x-quantifier



- *Deep Formula:*

$$((0 + U = U) \wedge (s(X) + Y = s(X + Y) \wedge s(X') + Y' = s(X' + Y')))$$

$$\Rightarrow s(Z) + (s(Z) + c) = s(s(c))$$

Orderings

- Structural ordering:
 - ▶ binary relation among the nodes in Δ
 - ▶ $\Delta_1 \prec_{\Delta} \Delta_2$ iff Δ_1 dominates Δ_2 in Δ .
- Quantifier ordering:
 - ▶ binary relation among the nodes in Δ
 - ▶ $\Delta_0 \prec_{\forall}^{\sigma} \Delta_1$ iff there is an $X \in \text{dom}(\sigma)$ bound on Δ_1 and in $\sigma(X)$ occurs a δ -variable bound on Δ_0 .
- Reduction relation: $\triangleleft := (\prec \cup \prec_{\forall})^+$
- A substitution is *admissible* if \triangleleft is irreflexive

Paths, Connections

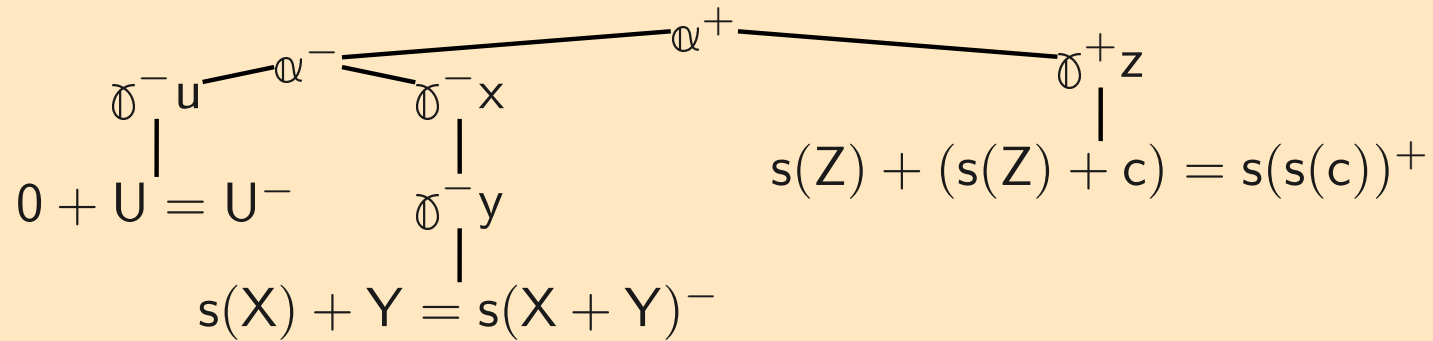
- Paths: $P \cup \{\ll \Gamma, \Delta' \gg\} \in \mathcal{P}(\Delta)$
 - (α -Dec) If $\Delta' := \alpha^p(\Delta_1, \Delta_2)$, then $P \cup \{\ll \Gamma, \Delta_1, \Delta_2 \gg\} \in \mathcal{P}(\Delta)$;
 - (β -Dec) If $\Delta' := \beta^p(\Delta_1, \Delta_2)$, then
 - $P \cup \{\ll \Gamma, \Delta_1 \gg, \ll \Gamma, \Delta_2 \gg\} \in \mathcal{P}(\Delta)$;
 - (γ -Dec) If $\Delta' := \gamma^p(\Delta_1, \dots, \Delta_n)$, then
 - $P \cup \{\ll \Gamma, \Delta_1, \dots, \Delta_n \gg\} \in \mathcal{P}(\Delta)$.
 - (δ -Dec) If $\Delta' := \delta^p(\Delta'')$, then $P \cup \{\ll \Gamma, \Delta'' \gg\} \in \mathcal{P}(\Delta)$.
- *Connection* = pair of subtrees of the *same* formula, *opposite* polarities and on a *same* path
- Allow connection between non-literal nodes using labels by automatic propagation to the literals
 - (technical details in the paper, p.55)

Extensional Expansion Proof Rules



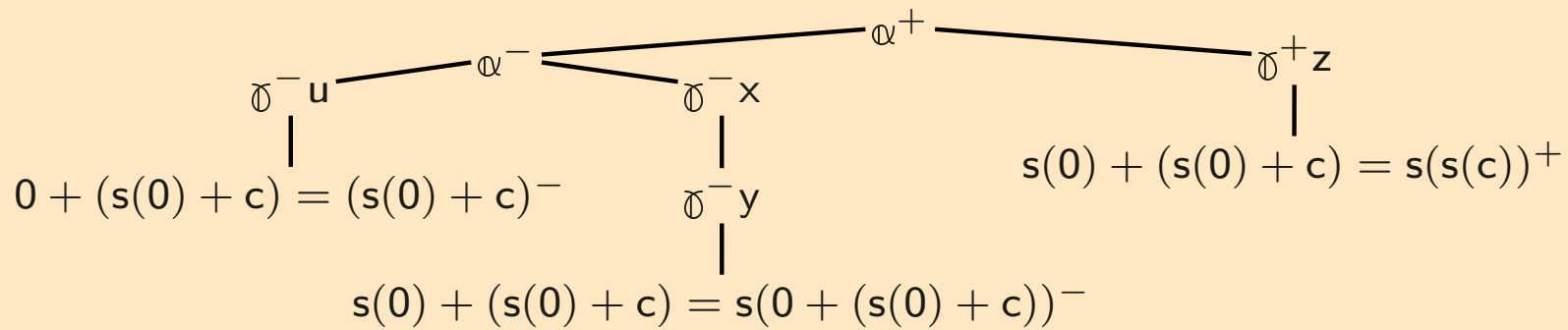
- Rule (bottom up reading): $\frac{\Delta'; \sigma' \triangleright \mathcal{C}'}{\Delta; \sigma \triangleright \mathcal{C}}$
- Rule is sound, if it preserves
 - ▶ admissibility of the substitution from σ to σ'
 - ▶ existence of satisfiable paths from Δ to Δ'
- Rule is safe, if it preserves
 - ▶ admissibility of the substitution from σ' to σ
 - ▶ existence of satisfiable paths from Δ' to Δ
- Rules for inserting connections, application of substitutions, Leibniz definition of equality, extensionality, cut.

Example Proof



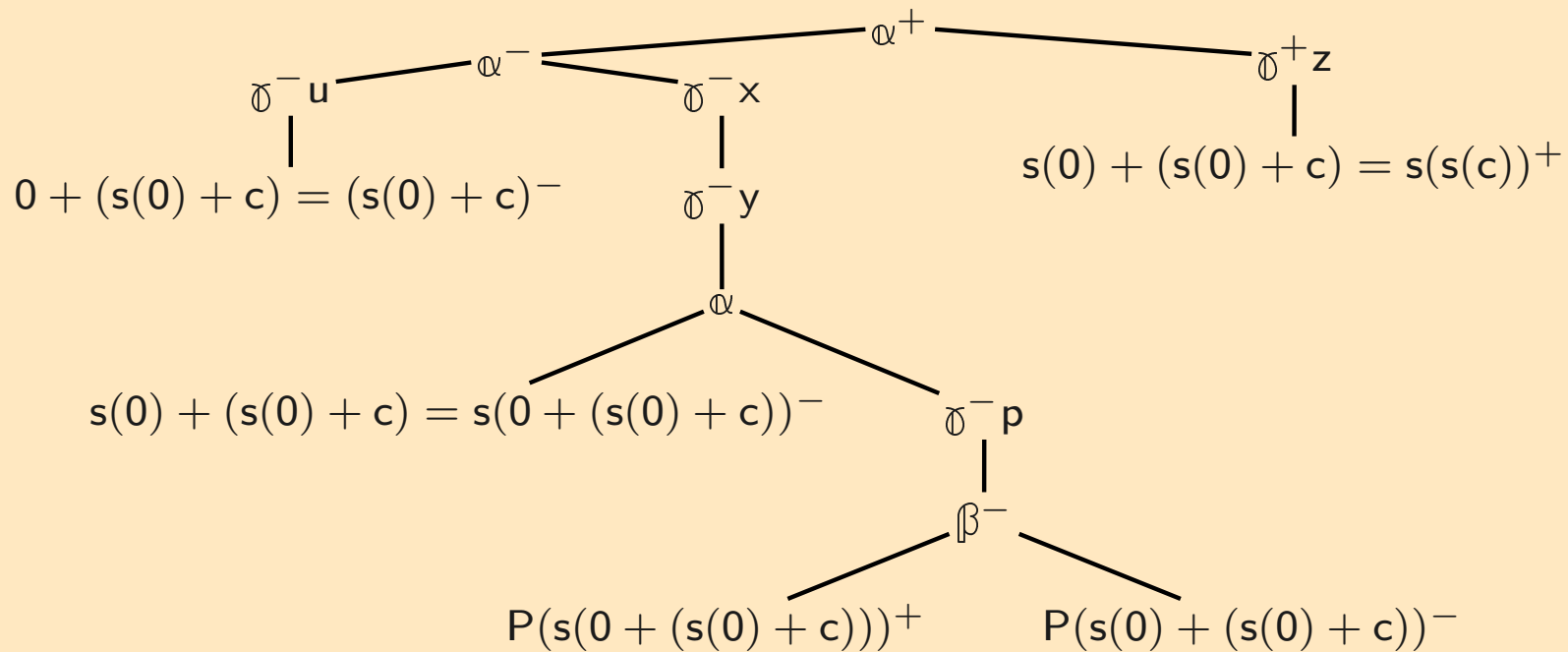
The initial IFT with multiplicity 1

Example Proof



Subst 0/X, Y/U, 0/Z, s(0) + c/Y

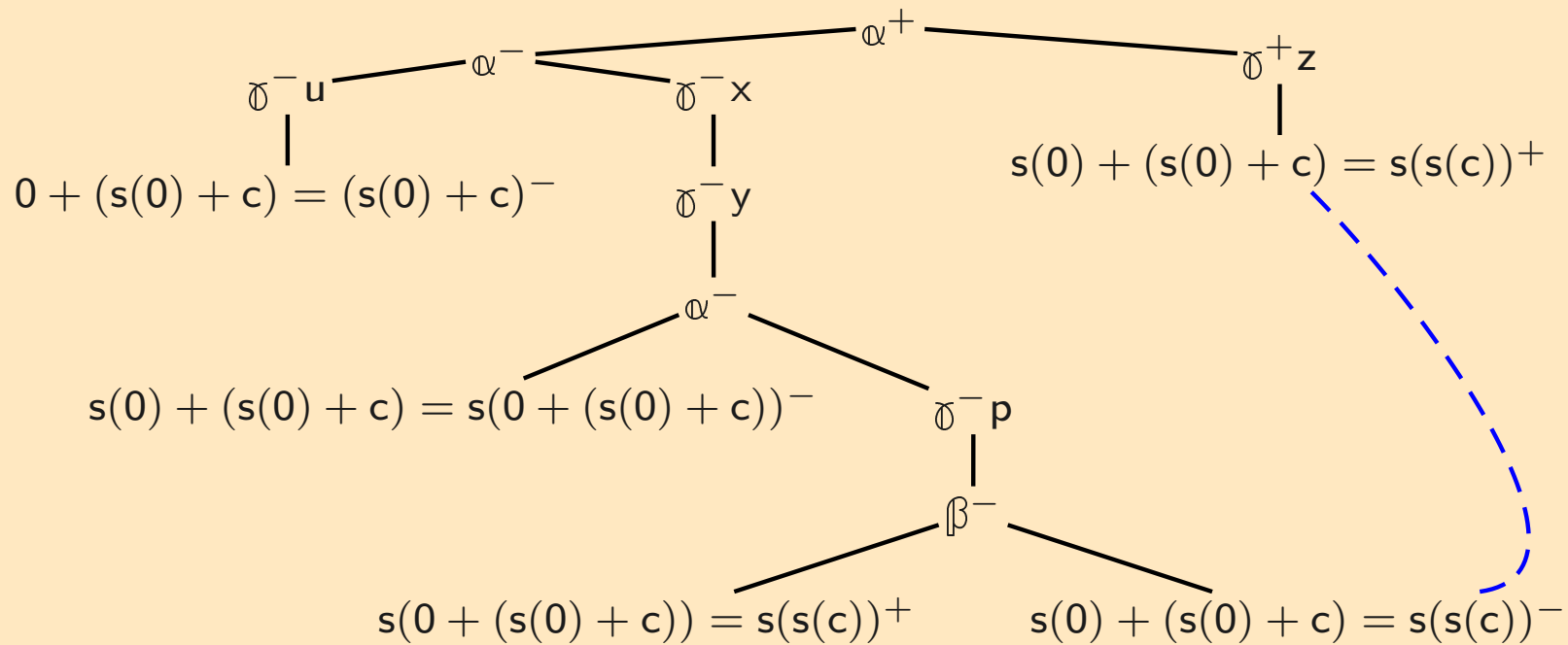
Example Proof



Expanding = into Leibniz' Definition of equality

$$s = t \rightarrow \forall P_{t \rightarrow o}. P(s) \Rightarrow P(t)$$

Example Proof



Add connection

But now the proof is stuck because all the initial free variable parts are instantiated...

What would we have liked to do to prevent this?

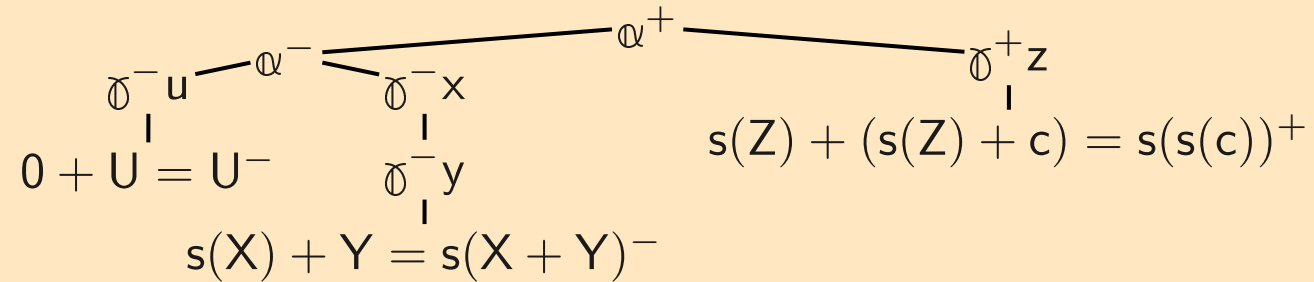


- Each time we instantiate a variable, increase the multiplicity of the quantifier this variable stems from

- Problem:
This is both doing
 1. too much (increase multiplicities unnecessarily)
 2. not enough to carry over connection information

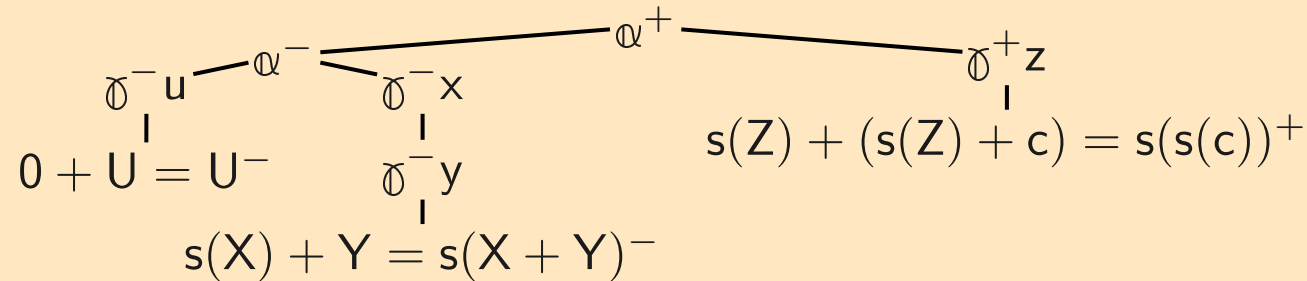
Why is it too much?

- Initial Tree:



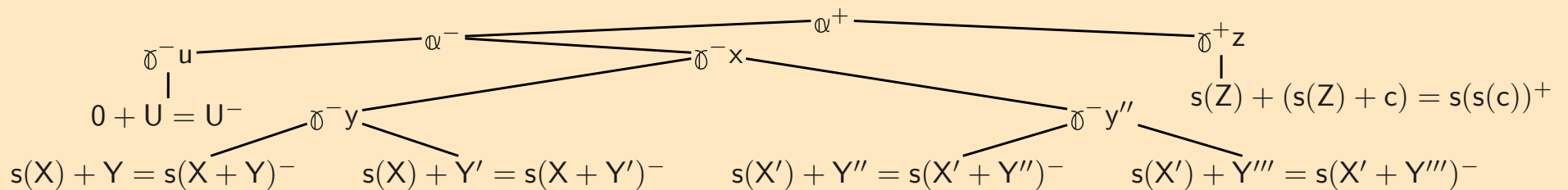
Why is it too much?

- Initial Tree:



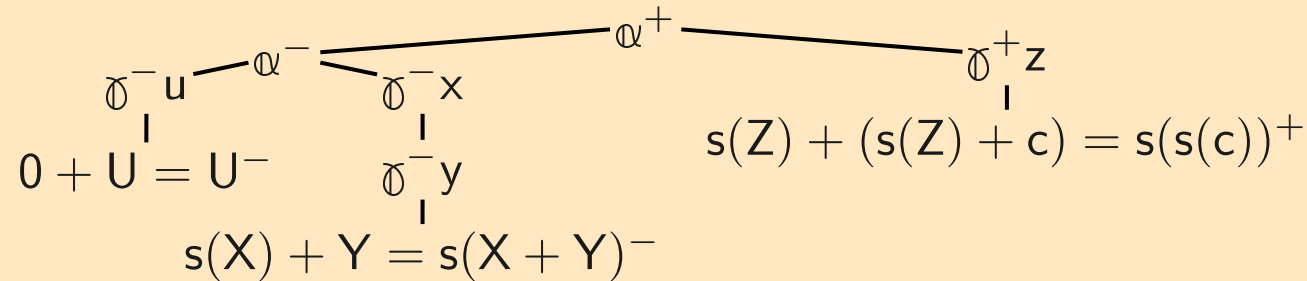
- If we increase the multiplicities of X and Y before *Subst*

$\sigma := \{0/X, s(0) + c/Y\}$ then we would have

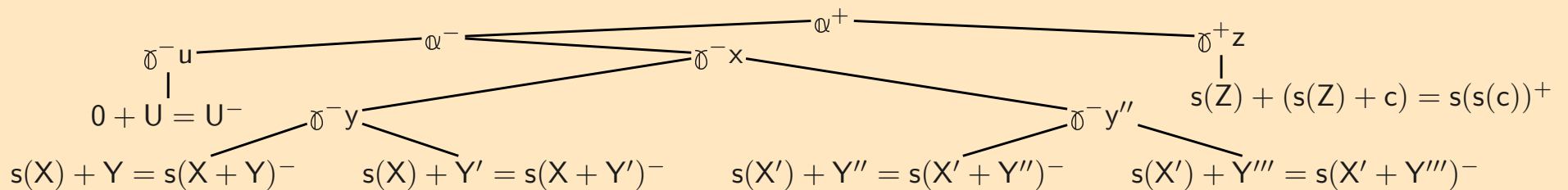


Why is it too much?

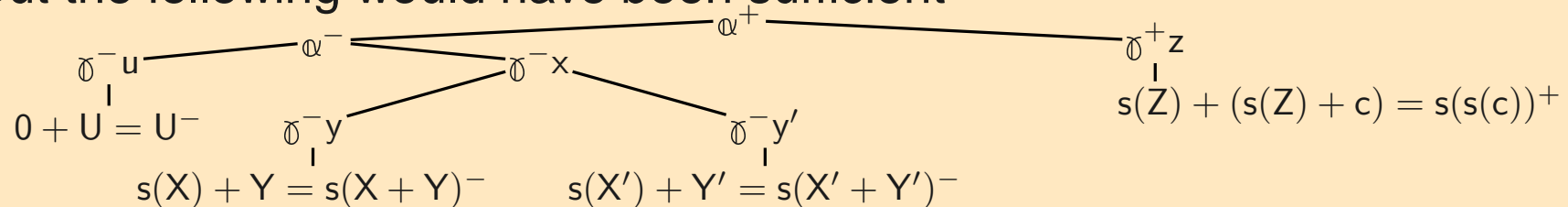
- Initial Tree:



- If we increase the multiplicities of X and Y before *Subst* $\sigma := \{0/X, s(0) + c/Y\}$ then we would have



- But the following would have been sufficient



Observation I

- If we want to increase the multiplicities of a set of quantifiers
- Only increase multiplicities of quantifiers that are *smallest wrt. structural ordering*,
The multiplicity of any other quantifier below them is implicitly increased

Why is it not enough to carry over connections?



- Consider:

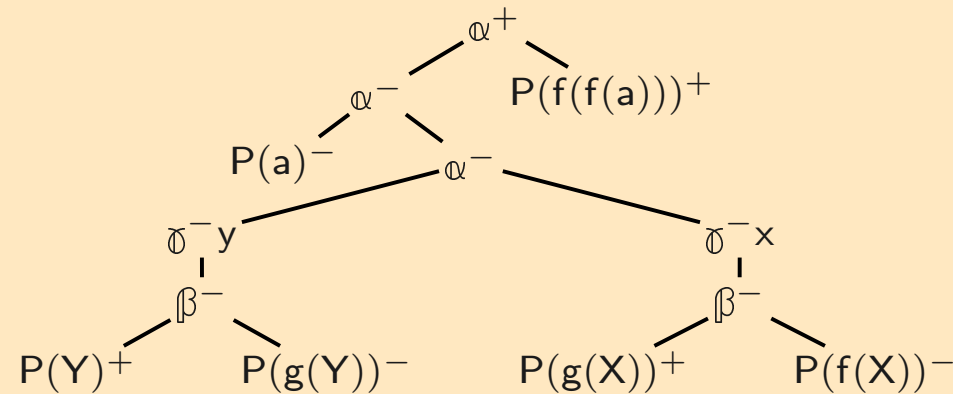
$$(P(a) \wedge (\forall x . P(x) \Rightarrow P(g(x)) \wedge \forall y . P(g(y)) \Rightarrow P(f(y)))) \Rightarrow P(f(f(a)))$$

Why is it not enough to carry over connections?



- Consider:

$$(P(a) \wedge (\forall x . P(x) \Rightarrow P(g(x)) \wedge \forall y . P(g(y)) \Rightarrow P(f(y)))) \Rightarrow P(f(f(a)))$$

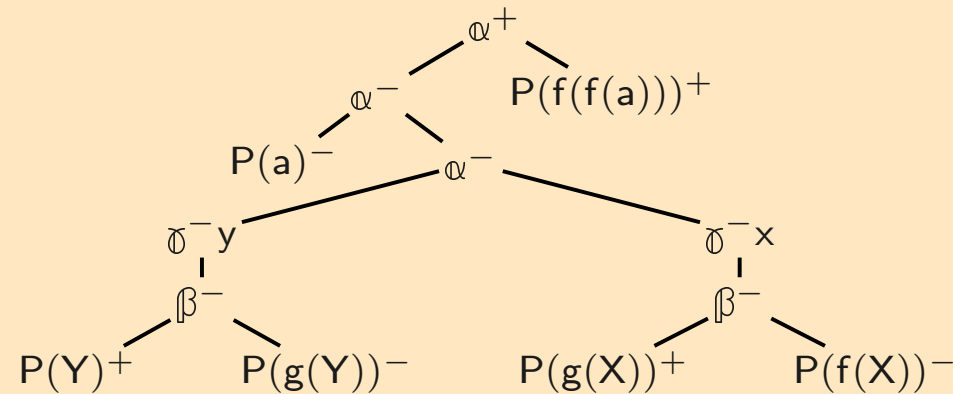


Why is it not enough to carry over connections?



- Consider:

$$(P(a) \wedge (\forall x . P(x) \Rightarrow P(g(x)) \wedge \forall y . P(g(y)) \Rightarrow P(f(y)))) \Rightarrow P(f(f(a)))$$



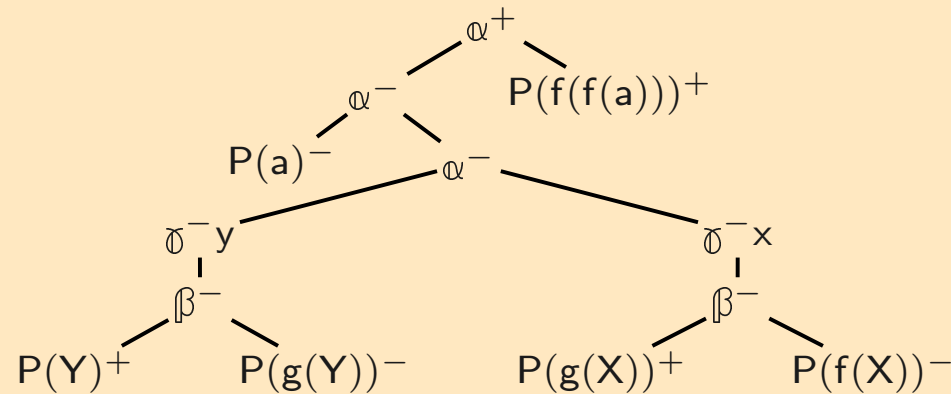
- Increase multiplicity of quantifier for Y and then substituting {X/Y}

Why is it not enough to carry over connections?

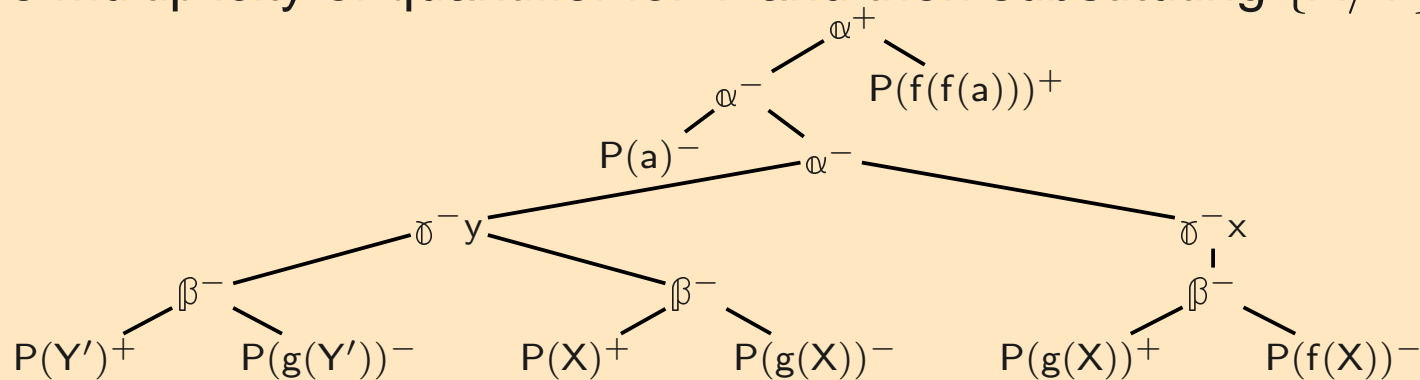


- Consider:

$$(P(a) \wedge (\forall x . P(x) \Rightarrow P(g(x)) \wedge \forall y . P(g(y)) \Rightarrow P(f(y)))) \Rightarrow P(f(f(a)))$$



- Increase multiplicity of quantifier for Y and then substituting {X/Y}

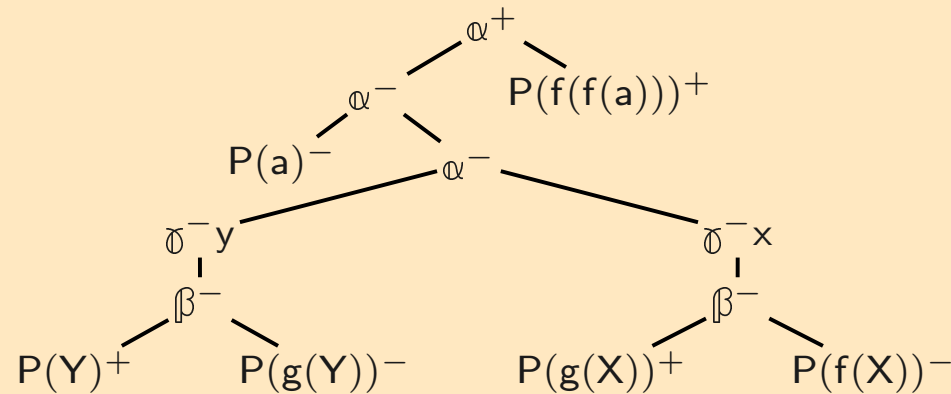


Why is it not enough to carry over connections?

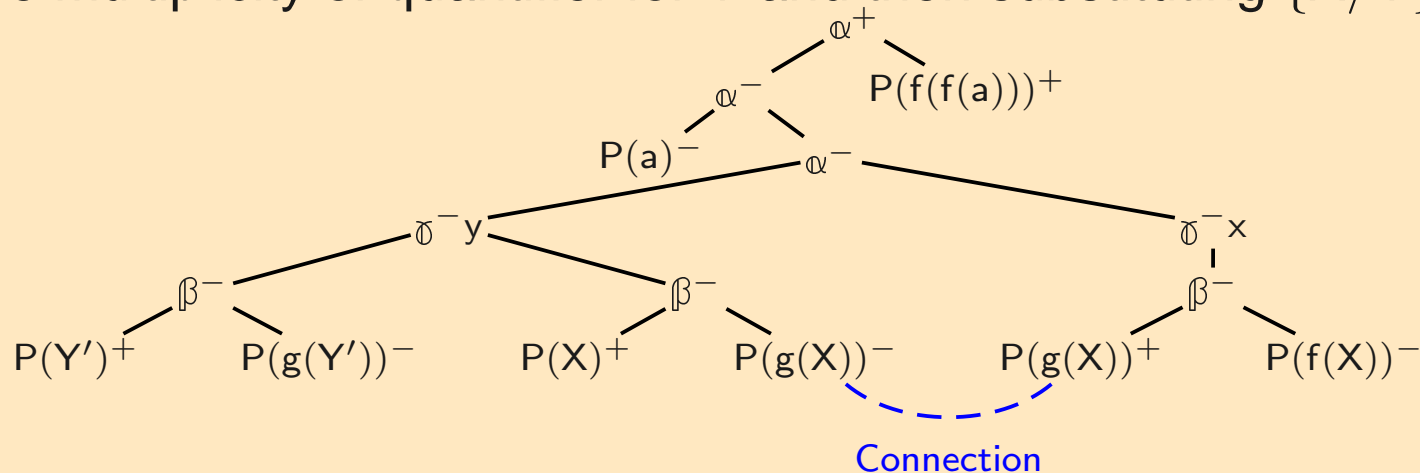


- Consider:

$$(P(a) \wedge (\forall x . P(x) \Rightarrow P(g(x)) \wedge \forall y . P(g(y)) \Rightarrow P(f(y)))) \Rightarrow P(f(f(a)))$$



- Increase multiplicity of quantifier for Y and then substituting {X/Y}

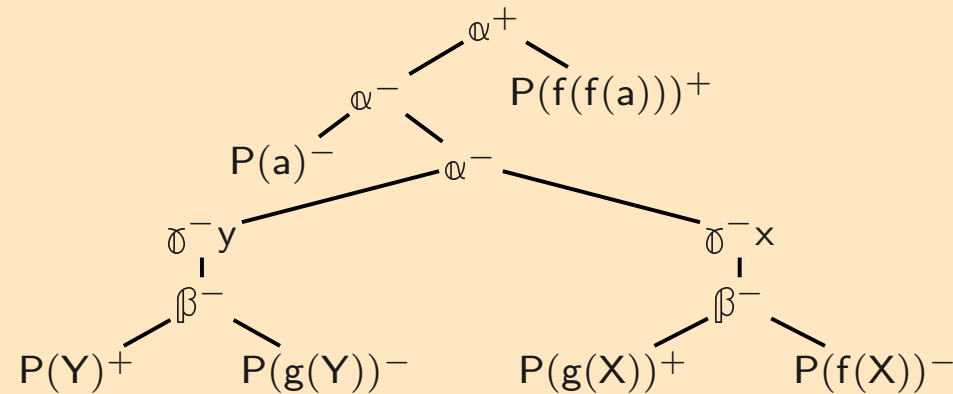


Why is it not enough to carry over connections?

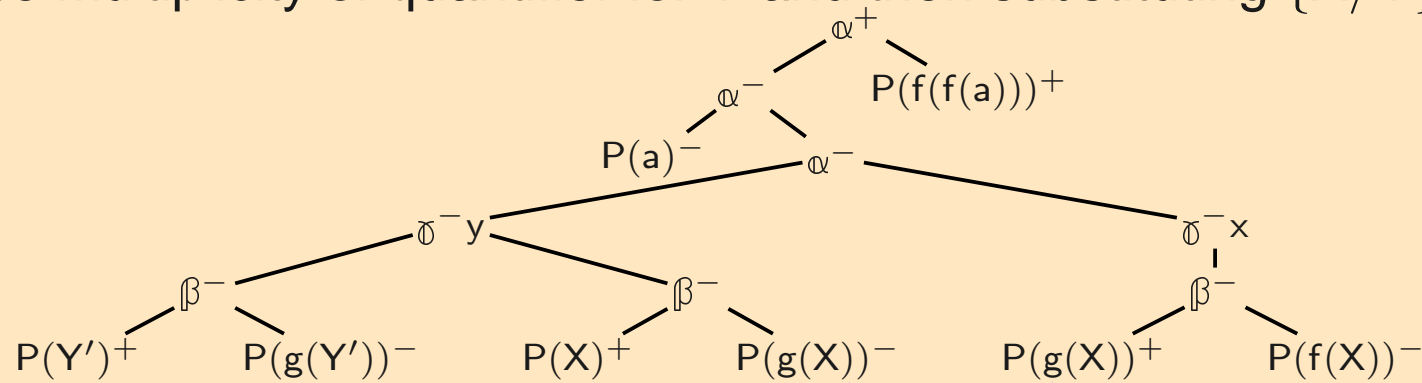


- Consider:

$$(P(a) \wedge (\forall x . P(x) \Rightarrow P(g(x)) \wedge \forall y . P(g(y)) \Rightarrow P(f(y)))) \Rightarrow P(f(f(a)))$$



- Increase multiplicity of quantifier for Y and then substituting {X/Y}



- Increase multiplicity of parent of X before applying subst {f(a)/X}

Loosing Connections

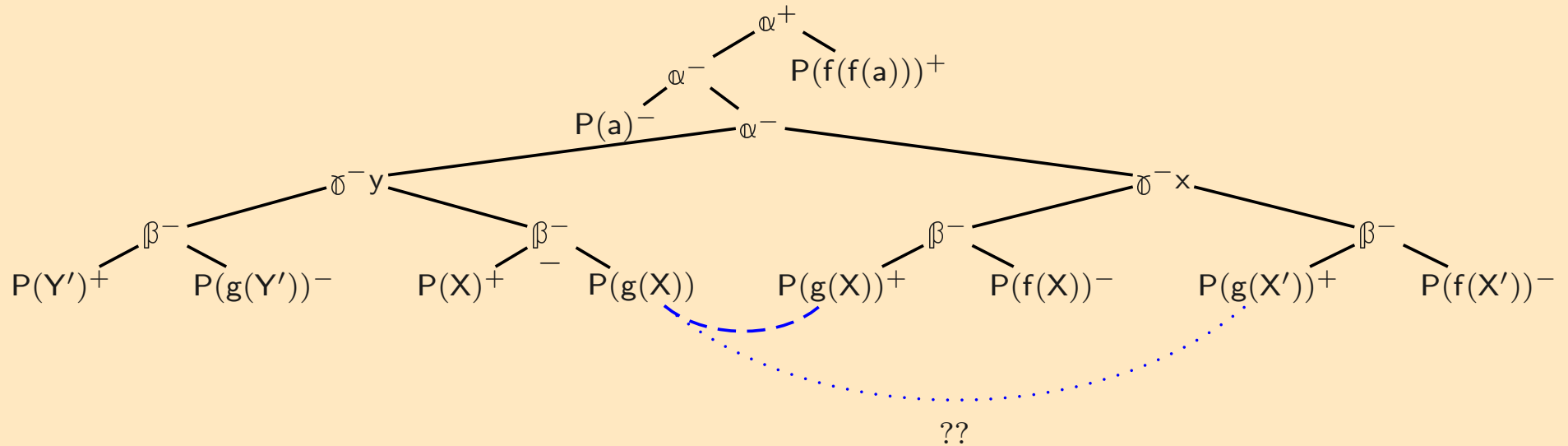


- Increase multiplicity of parent of X before applying subst $\{f(a)/X\}$

Loosing Connections

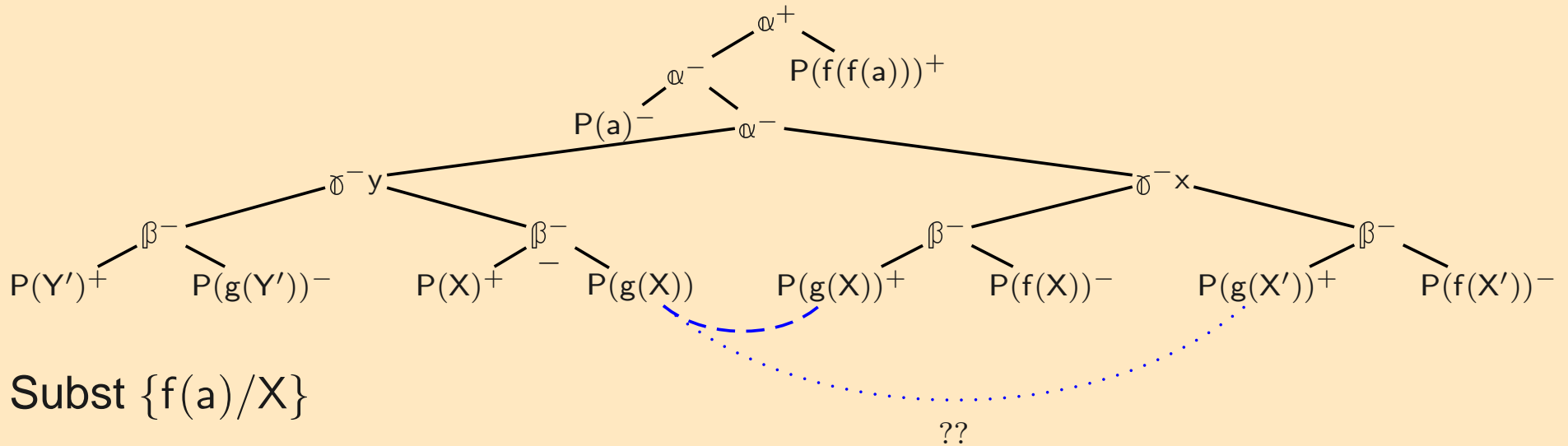


- Increase multiplicity of parent of X before applying subst $\{f(a)/X\}$



Loosing Connections

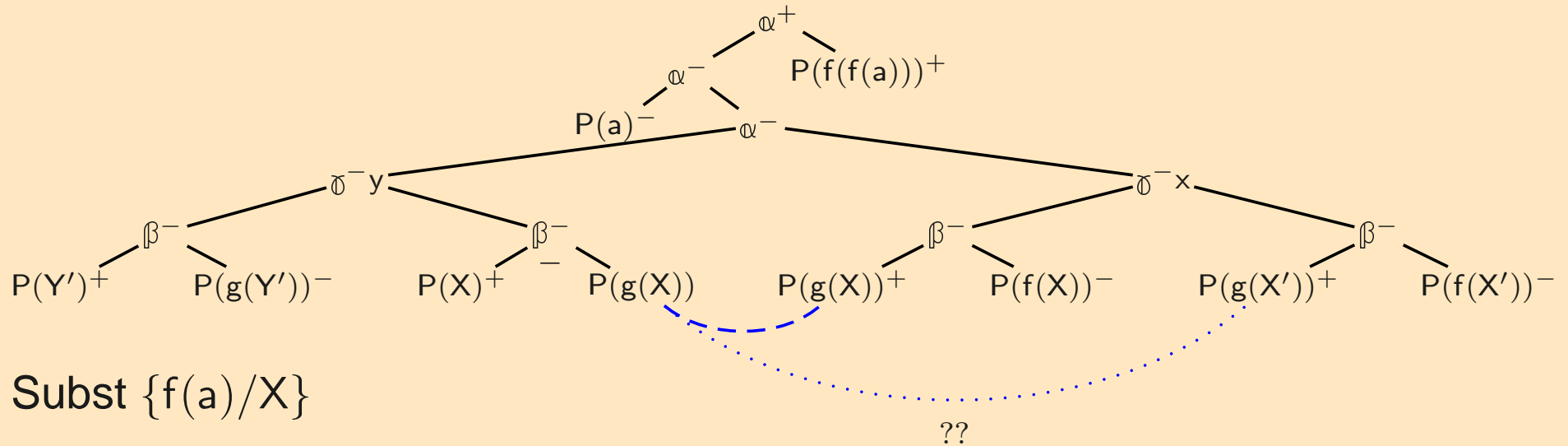
- Increase multiplicity of parent of X before applying subst {f(a)/X}



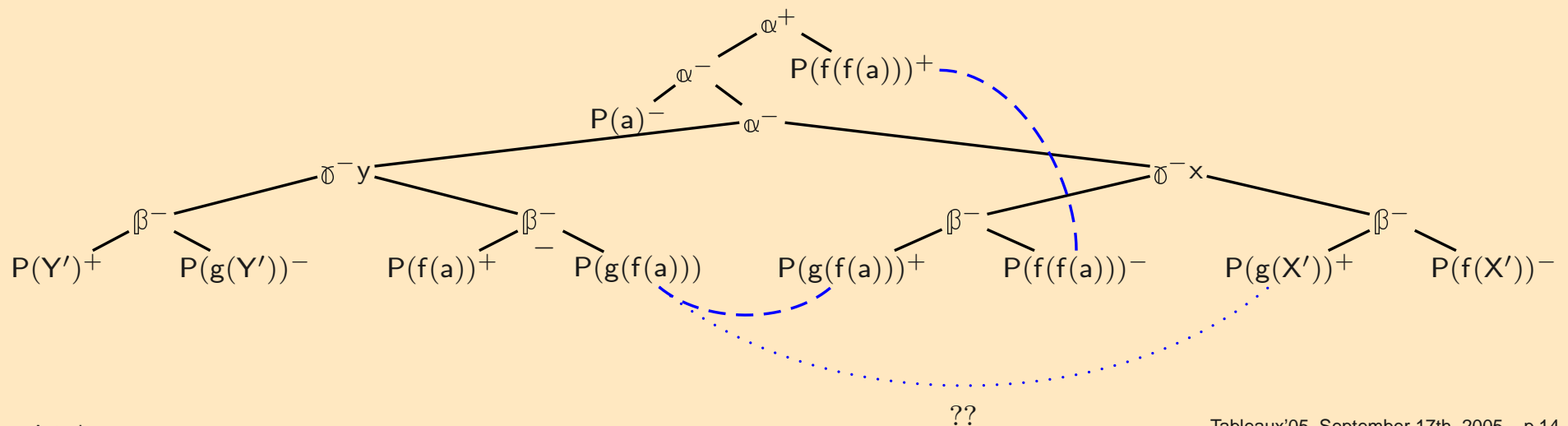
- Subst {f(a)/X}

Loosing Connections

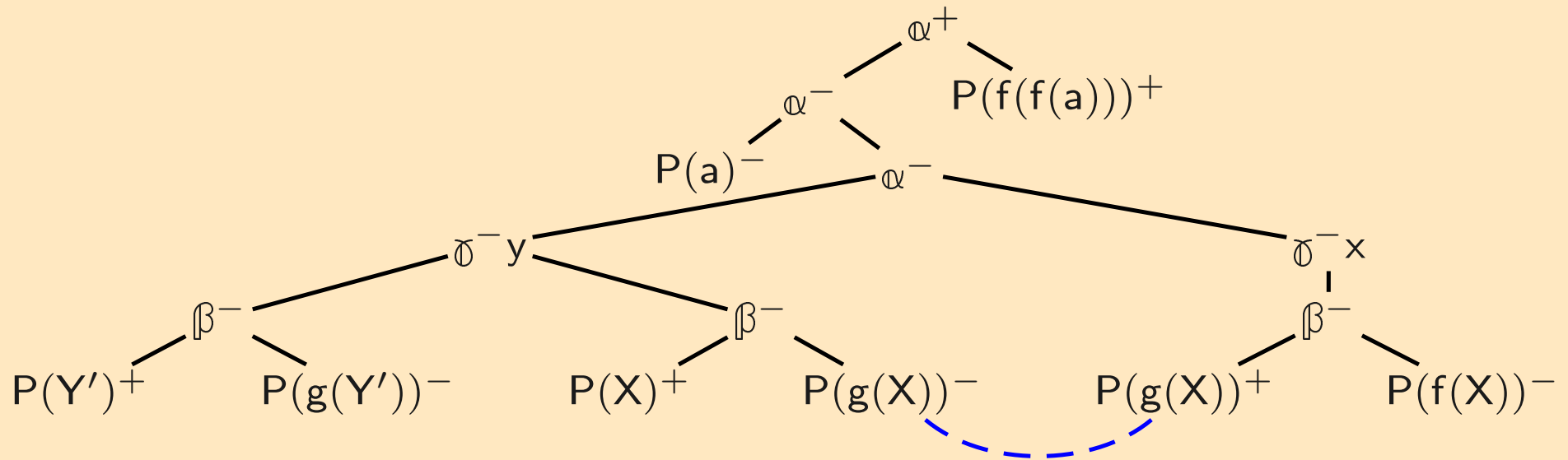
- Increase multiplicity of parent of X before applying subst $\{f(a)/X\}$



- Subst $\{f(a)/X\}$

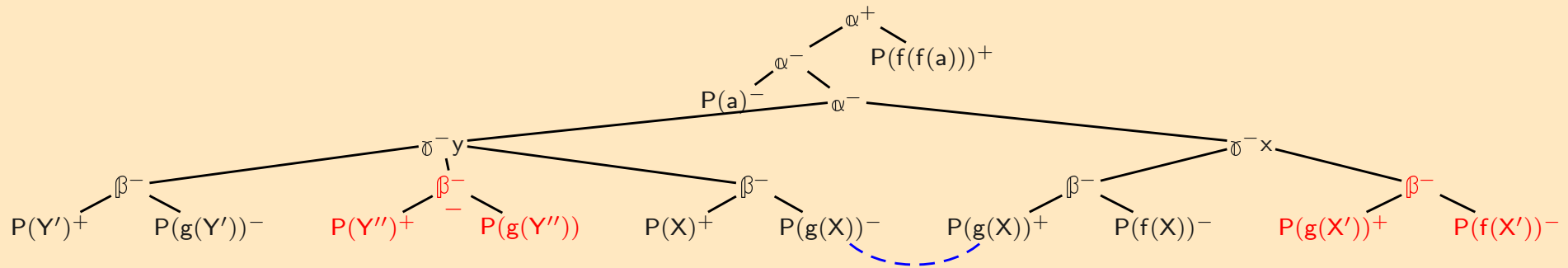


What we should have done...



- Current $\sigma := \{X/Y\}$

What we should have done...



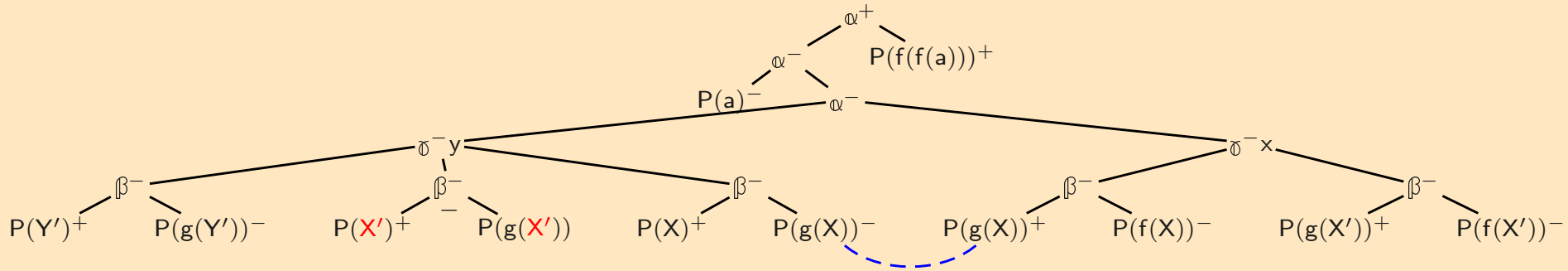
- Current $\sigma := \{X/Y\}$
- Increase multiplicity of parent of X *and parent of Y because X occurs in*
 $\sigma(Y) = X$

This gives (a) renaming $Y \rightarrow Y''$ and $X \rightarrow X'$ and

(b) isomorphic mapping of subtrees

(BTWO: $P(g(X))^- \rightarrow P(g(Y''))^-$ and $P(g(X))^+ \rightarrow P(g(X'))^+$)

What we should have done...



- Current $\sigma := \{X/Y\}$
- Increase multiplicity of parent of X *and parent of Y because X occurs in*
 $\sigma(Y) = X$

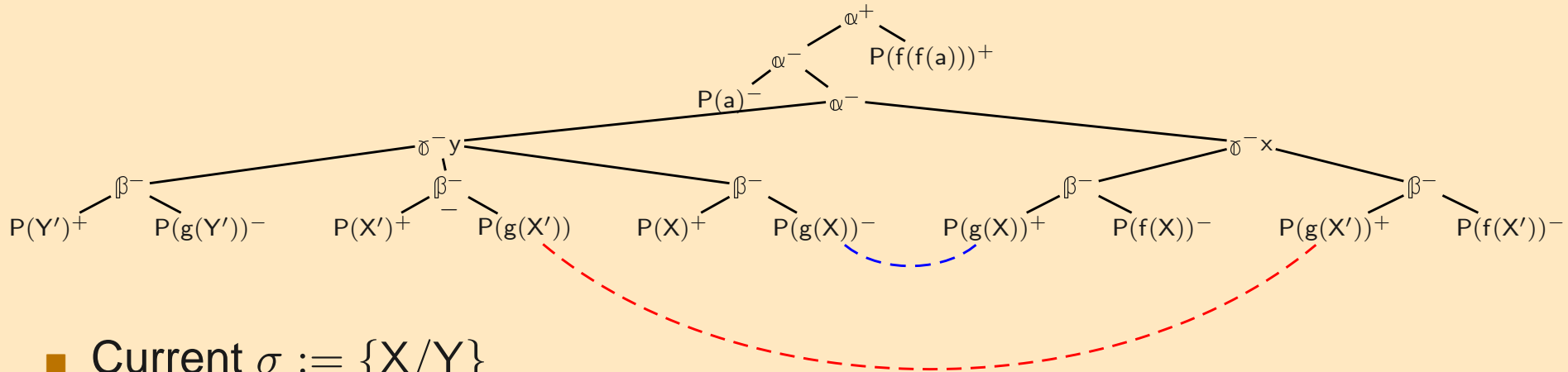
This gives (a) renaming $Y \rightarrow Y''$ and $X \rightarrow X'$ and

(b) isomorphic mapping of subtrees

(BTWO: $P(g(X))^- \rightarrow P(g(X'))^-$ and $P(g(X))^+ \rightarrow P(g(X'))^+$)

- Extend substitution using the renaming: from $\{X/Y\}$ get $\{X'/Y''\}$

What we should have done...



- Current $\sigma := \{X/Y\}$
- Increase multiplicity of parent of X *and parent of Y because X occurs in $\sigma(Y) = X$*

This gives (a) renaming $Y \rightarrow Y''$ and $X \rightarrow X'$ and

(b) isomorphic mapping of subtrees

(BTWO: $P(g(X))^- \rightarrow P(g(X'))^-$ and $P(g(X))^+ \rightarrow P(g(X'))^+$)

- Extend substitution using the renaming: from $\{X/Y\}$ get $\{X'/Y''\}$
- Carry over the connection using subtree mapping:
from $(P(g(X))^-, P(g(X))^+)$ get $(P(g(X'))^-, P(g(X'))^+)$

Observation II



- If we want to increase the multiplicity of the quantifier binding a variable X

Observation II

- If we want to increase the multiplicity of the quantifier binding a variable X
- We must increase increase the multiplicities of the quantifiers of those variables Y where X occurs in $\sigma(Y)$

Includes all variables and parameters in the subtree belonging to X

Observation II

- If we want to increase the multiplicity of the quantifier binding a variable X
- We must increase increase the multiplicities of the quantifiers of those variables Y where X occurs in $\sigma(Y)$

Includes all variables and parameters in the subtree belonging to X

- Not simply interested in increasing arbitrarily the multiplicity of a quantifier
But: We want a *copy* of the specific subtree of X

Observation II

- If we want to increase the multiplicity of the quantifier binding a variable X
- We must increase increase the multiplicities of the quantifiers of those variables Y where X occurs in $\sigma(Y)$

Includes all variables and parameters in the subtree belonging to X

- Not simply interested in increasing arbitrarily the multiplicity of a quantifier

But: We want a *copy* of the specific subtree of X

- From copying we obtain renamings and isomorphic mappings of subtrees to compute the new substitution and carry over the connections

Convex Set of Subtrees

- Definition 13 (Convex Set of Subtrees).** Let Δ be an extensional expansion tree, σ an admissible substitution, and \mathcal{K} a set of independent subtrees of Δ . \mathcal{K} is *convex with respect to σ* if, and only if, for all $\Delta' \in \mathcal{K}$ and for all γ - and δ -variables x bound in Δ' we have: if x occurs in some instance $\sigma(Y)$ for some Y , then there exists some $\Delta'' \in \mathcal{K}$ in which Y is bound.
- Lemma 2 (Maximality of Convex Sets of Subtrees).** Let Δ be an extensional expansion tree, σ an admissible substitution, and \mathcal{K} a convex set of subtrees of Δ . For all $x \in \text{dom}(\sigma)$, if x is not bound in \mathcal{K} , then all variables that occur in $\sigma(x)$ are also not bound in \mathcal{K} .

i.e. no nested subtrees.

Extended $\mathcal{E}\mathcal{E}\mathcal{P}$ rules

- Multiplicity increasing rule:

$$\frac{\Delta_{\overline{\sigma}^p X_1}(\dots, \Delta_{X_1}, \Delta'_{X'_1}, \dots), \dots, \Delta_{\overline{\sigma}^p X_n}(\dots, \Delta_{X_n}, \Delta'_{X'_n}, \dots); \sigma' \circ \sigma \triangleright \mathcal{C} \cup \iota(\mathcal{C})}{\Delta_{\overline{\sigma}^p X_1}(\dots, \Delta_{X_1}, \dots), \dots, \Delta_{\overline{\sigma}^p X_n}(\dots, \Delta_{X_n}, \dots); \sigma \triangleright \mathcal{C}} \mu\text{-Inc}$$

1. $\{\Delta_{X_1}, \dots, \Delta_{X_n}\}$ is a convex set with respect to σ
2. $\Delta'_{X'_i}$ are the copies of Δ_{X_i}
3. ρ and ι are respectively the renamings and the mapping of subtrees obtained from the copying, and
4. $\sigma' := [\rho(\sigma(X)) / \rho(X) \mid \text{for all } \gamma\text{-variables } X \in \text{dom}(\rho)]$.

- **Lemma 4. (Soundness & Safeness of Multiplicity Increase)** *The increase of multiplicities is sound and safe.*

Increase by Copying vs. Increase from Scratch



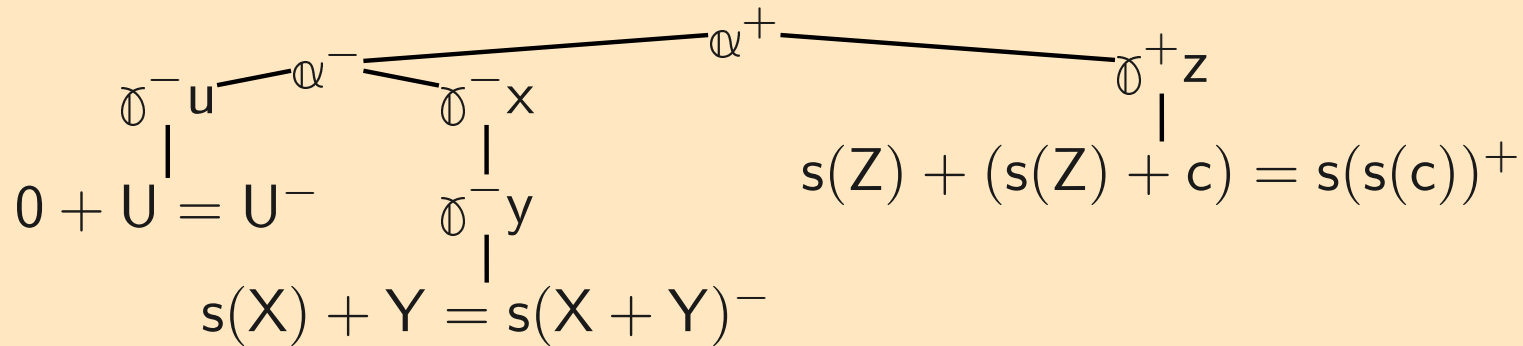
- Copying carries over all multiplicities and $=$ -rule applications applied in the subtrees Δ_{x_i}
- Increasing from scratch does not.

Soundness & Completeness

- **Theorem 2. (Soundness & Completeness of $\mathcal{EEP} + \mu\text{-Inc}$)** F is a valid formula if, and only if, there exists an extensional expansion proof using \mathcal{EEP} augmented by $\mu\text{-Inc}$ for $\Delta^{F^+, \mu_1}; id \triangleright \emptyset$, where μ_1 denotes the singular multiplicity.
- *Proof Idea.*
 - ▶ The substitution rule is the only unsafe rule
 - ▶ We can always combine substitution rule with the rule to increase the multiplicities such that both together form a safe proof step.
- Requires mechanism to determine
 - ▶ for a given set of variables that shall be substituted ($Domain(\sigma)$)
 - ▶ the subtrees ($\{\Delta_{x_1}, \dots, \Delta_{x_n}\}$) to increase the multiplicities (of the parents of these nodes)

technical, combines Observations I & II, paper, Def.15, p. 59

Example Proof Revisited

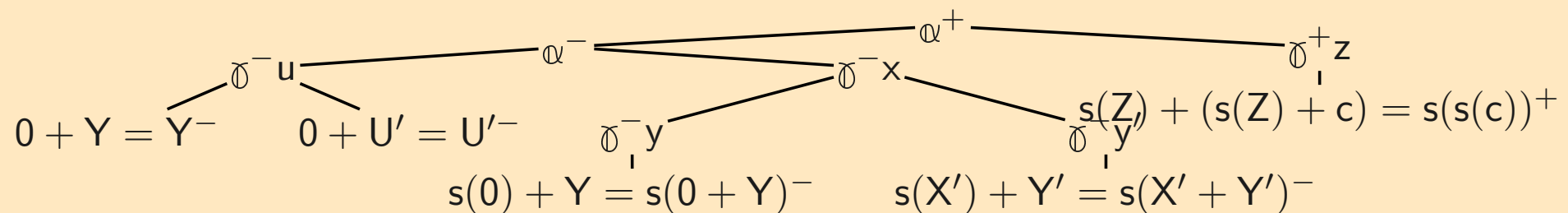


The initial IFT with multiplicity 1

Next: Apply $\{0/X, Y/U\}$

Requires: Increase of the multiplicities of the parents of X and U

Example Proof Revisited

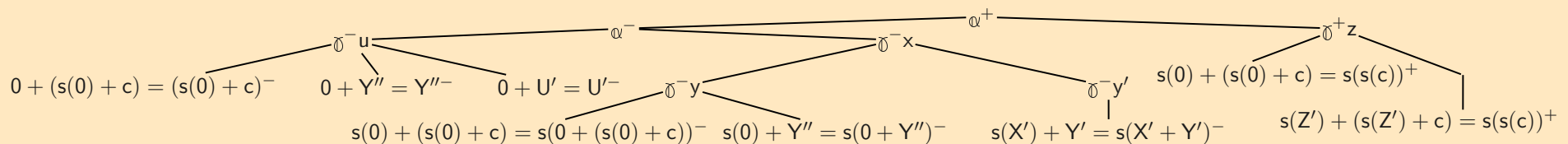


Increase of multiplicities for $\phi^- u$ and $\phi^- x$
 followed by $\{0/X, Y/U\}$

Next: apply $\{s(0) + c/Y, 0/Z\}$

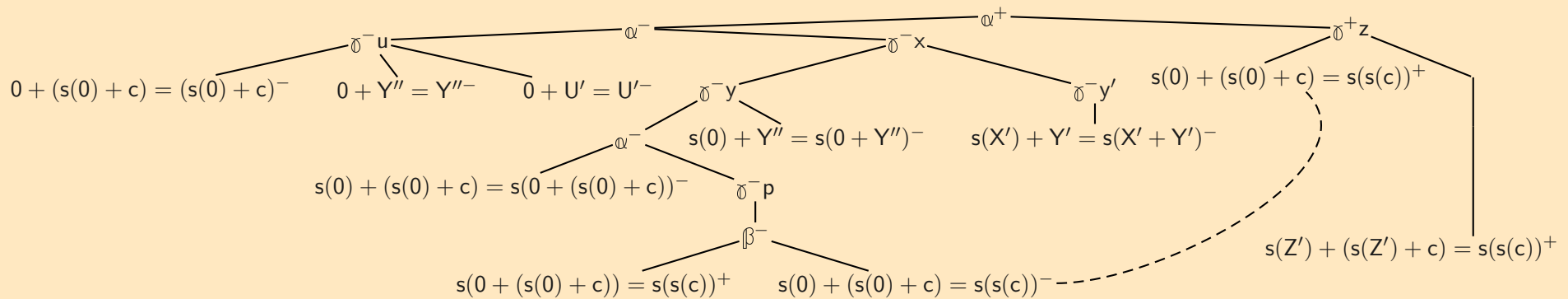
Requires: increase multiplicities of parents of Y and Z *and U*

Example Proof Revisited



- The overall substitution now is $\{0/X, s(0) + c/U, Y''/U'', s(0) + c/Y, 0/Z\}$
- Next: Introduce Leibniz' definition of equality and add connection

Example Proof Revisited



- Now the proof is not blocked and we can continue

Related Work



Sunil Issar in the context of the TPS system

[Issar90]

- Multiplicity increasing procedure for the JForms [An-81-a]
- Restricted to FOL
- Increases the multiplicities on demand for some problematic path for which no connection can be found.
- Copying and renaming a part of the JForm, but restricting the effect to the problematic path in order to control the increase of the number of new paths.
- Does not adapt existing substitutions and connections
- Complete as substitutions are never applied to the JForm (probably the major reason for restriction to first-order logic)

Conclusion

- Expansion proof calculus for HOL
- With a rule to dynamically increase multiplicities
- Avoids restarts
- Preserves connections of copied parts
- Can be combined with the substitution rule to make substitution applications safe
- Can be adapted for the modal logics considered by Wallen (1990)

[Autexier03]