

An alternative way of using unification in proof-search

1. Notions and conventions

- *Parameters* are free variables, never used in quantification; discourse-parameters for parameters: $a, b \dots$
- *Term-namers* (or *namers* for short) are special identifiers; discourse-parameters for namers: $X, Y \dots$
- *Super-terms* are generated from constants, parameters and namers, using the vocabulary's function identifiers.
- An *equation-constraint (EC)* is an equation $t = t'$ between super-terms. This EC is *basic* if it is of the form $X = t$, with X not in t .
- An *independence-constraint (IC)* is an expression $a \notin t$ where a is a parameter and t a super-term. An IC $a \notin X$ is said to be *basic*.
- We refer to ECs and ICs jointly as *constraints*. A *constraint-set* is a finite set of constraints. Discourse-parameters for constraint-sets: K, \dots A constraint set is *basic* if it consists only of basic ICs (and no ECs!).
- If \vec{t} is $(t_1 \dots t_k)$ and \vec{s} is $(s_1 \dots s_k)$, then
 - $\vec{t} = \vec{s}$ stands for $t_1 = s_1, \dots, t_k = s_k$, and
 - $a \notin \vec{t}$ for $a \notin t_1, \dots, a \notin t_k$.

2. Logic Inference Rules

We define a sequent calculus **D** whose rules are correct *modulo a constraint-set* K .

- The proposition rules are as usual.

$$\exists L \quad \frac{\Gamma, A[a] \Rightarrow B}{\Gamma, \exists x A[x] \Rightarrow B} \quad \begin{array}{l} a \text{ not in the derived sequent, and} \\ \text{for each } X \text{ occurring here, the basic IC } a \notin X \text{ is in } K. \end{array}$$

$$\exists R \quad \frac{\Gamma \Rightarrow A[X]}{\Gamma \Rightarrow \exists x A[x]} \quad X \text{ fresh}$$

$$\forall L \quad \frac{\Gamma, \forall x A[x], A[X] \Rightarrow B}{\Gamma, \forall x A[x] \Rightarrow B} \quad X \text{ fresh}$$

$$\forall R \quad \frac{\Gamma \Rightarrow A[a]}{\Gamma \Rightarrow \forall x A[x]} \quad \begin{array}{l} a \text{ not in the derived sequent, and} \\ \text{for each } X \text{ occurring here, the IC } a \notin X \text{ is in } K. \end{array}$$

Initial

$$\frac{}{\Gamma, P(\vec{t}) \Rightarrow P(\vec{s})} \quad \text{provided the equations } \vec{t} = \vec{s} \text{ are in } K$$

3. Constraint inference rules

Note: The intent is that a constraint-set K is “true” when there is a substitution $\theta = \{t_1 \dots t_k / X_1 \dots X_k\}$ of terms for namers, so that $\theta(t)$ is identical to $\theta(t')$ for every equation $t = t'$ in K , and if an IC $a \notin X_i$ is in K , then the parameter a does not occur in t_i .

Constraint trimming

$$\frac{K}{K, t = t}$$

Equation decomposition

$$\frac{K, \vec{t} = \vec{s}}{K, f(\vec{t}) = f(\vec{s})}$$

Instantiation

$$\frac{\{t/X\} K}{K, X = t} \quad \frac{\{t/X\} K}{K, t = X} \quad X \text{ not in } t$$

Note that the substitution applies also to occurrences of X in ICs.

Independence decomposition

$$\frac{K, a \notin \vec{t}}{K, a \notin f(\vec{t})}$$

For $k = 0$, i.e. when f is a constant identifier, we have the special case:

$$\frac{K}{K, a \notin f}$$

Independence trimming

$$\frac{K}{K, a \notin b} \quad (a \text{ and } b \text{ distinct parameters})$$

Basic constraint-set

$$\frac{}{K} \quad K \text{ basic}$$

4. Examples

1. Let K consist of the equation $X = Y$. The following derivation is correct modulo K :

$$\frac{\frac{\frac{P(X) \Rightarrow P(Y)}{\forall x P(x) \Rightarrow P(Y)}}{\forall x P(x) \Rightarrow \exists y P(y)}}{\Rightarrow \forall x P(x) \rightarrow \exists y P(y)}$$

The constraint-set K is derivable:

$$\frac{\emptyset}{X = Y}$$

by an instance of the Namer-Instantiation Rule.

2. The following derivation is correct modulo the constraint-set $K = \{b = V, Y = a\}$.

$$\frac{\frac{\frac{\frac{P(b, Y) \longrightarrow P(V, a)}{\forall y P(b, y) \longrightarrow P(V, a)}}{\forall y P(b, y) \longrightarrow \exists v P(v, a)}}{\exists x \forall y P(x, y) \longrightarrow \exists v P(v, a)}}{\frac{\exists x \forall y P(x, y) \longrightarrow \forall u \exists v P(v, u)}}{\Rightarrow \exists x \forall y P(x, y) \rightarrow \forall u \exists v P(v, u)}}$$

The constraint-set K is trivially derivable.

3. The following derivation is correct modulo the constraint-set $K = \{X = ga, Y = fX, fY = ffgZ\}$.

$$\begin{array}{c}
\frac{P(ga) \Rightarrow P(X) \quad P(fX), P(ga) \Rightarrow P(Y)}{P(X) \rightarrow P(fX), P(ga) \Rightarrow P(Y)} \quad \frac{P(ga) \Rightarrow P(X) \quad P(fY), P(fX), P(ga) \Rightarrow P(ffgZ)}{P(fY), (P(X) \rightarrow P(fX)), P(ga) \Rightarrow P(ffgZ)} \\
\hline
\frac{P(Y) \rightarrow P(fY), (P(X) \rightarrow P(fX)), P(ga) \Rightarrow P(ffgZ)}{\forall x(P(x) \rightarrow P(fx)), (P(X) \rightarrow P(fX)), P(ga) \Rightarrow P(ffgZ)} \\
\hline
\frac{\forall x(P(x) \rightarrow P(fx)), P(ga) \Rightarrow \exists z P(ffgz)}{(\forall x(P(x) \rightarrow P(fx))), (\exists y P(gy)) \Rightarrow \exists z P(ffgz)} \\
\hline
\Rightarrow (\forall x(P(x) \rightarrow P(fx))) \wedge (\exists y P(gy)) \rightarrow \exists z P(ffgz)
\end{array}$$

A derivation for K :

$$\begin{array}{c}
\frac{\emptyset}{a = Z} \\
\frac{a = Z}{ga = gZ} \\
\frac{ga = gZ}{fga = ffgZ} \\
\frac{fga = ffgZ}{ffga = ffgZ} \\
\hline
\frac{Y = fga, fY = ffgZ}{X = ga, Y = fX, fY = ffgZ}
\end{array}$$

4. The following derivation is correct modulo the constraint-set $K = \{X = W, b = Y, Y = Z, W = a, b \notin X\}$:

$$\begin{array}{c}
\frac{P(X, b) \Rightarrow P(W, Y) \quad P(Y, W) \Rightarrow P(Z, a)}{P(X, b), P(W, Y) \rightarrow P(Y, W) \Rightarrow P(Z, a)} \\
\frac{\frac{P(X, b), \forall x, y (P(x, y) \rightarrow P(y, x)) \Rightarrow \exists y P(y, a)}{\exists y P(X, y), \forall x, y (P(x, y) \rightarrow P(y, x)) \Rightarrow \exists y P(y, a)}}{\forall x \exists y P(x, y), \forall x, y (P(x, y) \rightarrow P(y, x)) \Rightarrow \exists y P(y, a)} \\
\frac{\forall x \exists y P(x, y), \forall x, y (P(x, y) \rightarrow P(y, x)) \Rightarrow \forall x \exists y P(y, x)}{\Rightarrow \forall x \exists y P(x, y) \wedge (\forall x, y (P(x, y) \rightarrow P(y, x))) \rightarrow \forall x \exists y P(y, x)}
\end{array}$$

A derivation for K :

$$\begin{array}{c}
\frac{b \notin a}{X = a, b \notin X} \\
\frac{b = Y, X = a, b \notin X}{b = W, W = Y, X = a, b \notin X} \\
\frac{b = W, W = Y, X = a, b \notin X}{X = Z, b = W, W = Y, Z = a, b \notin X}
\end{array}$$

5. Non-examples

1. The following derivation is correct modulo $K = \{a = b\}$. However, K is not derivable.

$$\frac{P(a) \Rightarrow P(b)}{\frac{}{(\exists x P(x)) \rightarrow (\forall y P(y))}}$$

2. The following derivation is correct modulo $K = \{a = Z, b = Z\}$, which is underivable, as it reduces to $\{a = b\}$.

$$\frac{\frac{P(a) \Rightarrow P(Z) \quad Q(b) \Rightarrow Q(Z)}{P(a), Q(b) \Rightarrow P(Z) \wedge Q(z)}}{\frac{}{\exists x P(x), \exists y Q(y) \Rightarrow \exists z P(z) \wedge Q(z)}}$$

3. The following derivation is correct modulo $K = \{X = b, a = U, a \notin X, b \notin X, b \notin U\}$. K is underivable, as it reduces to the constraint-set $\{b \notin b\}$.

$$\frac{\frac{\frac{P(X, a) \Rightarrow P(b, U)}{P(X, a) \Rightarrow \forall v P(v, U)}}{P(X, a) \Rightarrow \exists u \forall v P(v, u)}}{\frac{\exists y P(X, y) \Rightarrow \exists u \forall v P(v, u)}}{\forall x \exists y P(x, y) \Rightarrow \exists u \forall v P(v, u)}}$$

6. Constraint algorithmics.

If K is a constraint-set, a *substitution for K* is an assignment θ of a super-term t_i to each X_i in a listing \vec{X} of the namers in K . We write $\{\vec{t}/\vec{X}\}$ for θ . The result $\theta(t)$ of applying θ to a term t is defined as usual by recurrence on t . If every t_i above is a term, we say that the substitution θ is *namer-free*. The composition $\theta \circ \theta'$ of substitutions is said to be a *refinement* of θ . θ is a *most-general-substitution (MGS)* for a set Θ of substitutions if every element of Θ is a refinement of θ .

A substitution θ *satisfies* K if for every equation $t = t'$ in K the super-terms $\theta(t)$ and $\theta(t')$ are identical, and for every IC $a \notin t$ in K the parameter a does not occur in $\theta(t)$.

The inference rules for constraints are the core of an algorithm to resolve constraint-sets: given a constraint-set K , the algorithm either yields a substitution for K , or a failure message indicating that no such substitution exists.

The algorithm operates on *constraint-substitution pairs* $(K; S)$, where K is a constraint-set, and S a finite set of ECs of the form $X = t$ (intended to represent the substitution $\{t/X\}$). The pair is *terminal* if K is basic.

The algorithm applies nondeterministically the following reduction steps.

Constraint trimming:	$(K, t = t; S)$	\mapsto	$(K; S)$
Equation decomposition:	$(K, f(\vec{t}) = f(\vec{s}); S)$	\mapsto	$(K, \vec{t} = \vec{s}; S)$
Instantiation:	$(K, X = t; S)$	\mapsto	$(\{t/X\}K; \{t/X\}S, X = t)$
	$(K, t = X, S)$	\mapsto	$(\{t/X\}K; \{t/X\}S, X = t)$ (X not in t)
Independence decomposition	$(K, a \notin f(\vec{t}); S)$	\mapsto	$(K, a \notin \vec{t}; S)$
Independence trimming	$(K, a \notin b; S)$	\mapsto	$(K; S)$ (a and b distinct parameters)
Substitution failure	$(K, X = t; S)$	\mapsto	$(; 0 = 1)$
	$(K, t = X; S)$	\mapsto	$(; 0 = 1)$ (X in t)
Independence failure	$(K, a \notin t; S)$	\mapsto	$(; 0 = 1)$ (a in t)

Thus, no reduction applies to a terminal pair.

Lemma 1. *All executions of the algorithm terminate with a terminal pair.*

Proof. Main induction on the number of variables in $(K; S)$, and secondary induction on the size (i.e. number of symbols) of K . \dashv

We say that a substitution θ *satisfies* a pair $(K; S)$ if it satisfies K and is a refinement of S .

Lemma 2. *If $(K; S) \mapsto (K'; S')$ then a substitution θ satisfies $(K; S)$ iff it satisfies $(K'; S')$.* \dashv

Combining the two Lemmas we conclude:

Proposition 3. *If a constraint-set K is satisfied by θ , then $(K; \emptyset) \mapsto^* (K_0; S)$ where K_0 is basic and θ is a refinement of S . If K is not satisfiable, then $(K; \emptyset) \mapsto^* (\emptyset; 0 = 1)$.*

Soundness and Completeness.

Soundness Theorem. *If a sequent $\Gamma \Rightarrow A$ is derived in \mathbf{D} by a derivation \mathcal{D} modulo a derivable constraint-set, then it is derivable in the usual sequential calculus \mathbf{S} (with namers treated as free variables).*

Completeness Theorem. *If a sequent $\Gamma \Rightarrow A$ is derived in the usual sequential calculus, then there is a derivable constraint-set K such that $\Gamma \Rightarrow A$ is derivable modulo K . +*