A Isabelle definitions and theorems

This document describes proofs in Isabelle of some results relevant to the paper A Proof Theoretic Analysis of Intruder Theories. It contains proofs formulated for the system for Dolev-Yao intruders considered in Section 6, although the proofs include cut-admissibility and the existence of normal derivations, which are given in the paper for more complex theories, in Sections 3 and 4.

The proofs may be found in the files `Intruder.{thy,ML}` and `Int_DC.{thy,ML}` in `http://users.rsise.anu.edu.au/~jeremy/isabelle/2005/spi/`.

A.1 Definitions of intruder deduction

We present proofs for the system for Dolev-Yao intruders considered in Section 6, where the only constructors are message pairing and symmetric encryption. Note that the derivation rules are expressed *not* to include a cut rule. Some of the proofs described here are adapted from earlier work which dealt with deducibility in regard to observer theories (involving *pairs* of messages).

We setup the type of messages in Isabelle using a `datatype`:

```
datatype 'n msg = Name "'n" | Rigid nat
  | Mpair "'n msg" "'n msg" | Enc "'n msg" "'n msg"
```

The names of these constructors represent something of a historical anomaly: messages of the form `Name` x are variables (in some previous work called "free names", or just "names"), whereas messages of the form `Rigid` n are names (in some previous work called "rigid names").

For the Dolev-Yao intruders we use the sequent system shown in Figure 4. In Isabelle we define deducibility under these rules by an inductively defined set `dyder`. By way of example, we show the Isabelle code for the rule p_R.

```
"[| (?S, ?M) : dyder; (?S, ?N) : dyder |] ==> (?S, Mpair ?M ?N) : dyder"
```

Then $\Gamma \Vdash_{\mathcal{R}} M$ means that $\Gamma \vdash M$ is derivable using only the right rules p_R and e_R and the *id* rule; we can then define a normal, or linear, derivation using the rules shown in Figure 5.

In Isabelle the set of right-derivable sequents (normally derivable) sequents is `dyder_rt` (`dyder_norm`).

$$\frac{\Gamma \Vdash_{\mathcal{R}} M}{\Gamma \vdash M}\, id \qquad \frac{\Gamma, \langle M, N\rangle, M, N \vdash T}{\Gamma, \langle M, N\rangle \vdash T}\, p_L \qquad \frac{\Gamma, \{M\}_K \Vdash_{\mathcal{R}} K \quad \Gamma, \{M\}_K, M, K \vdash T}{\Gamma, \{M\}_K \vdash T}\, e_L$$

Fig. 5. Sequent system for normal derivations for Dolev-Yao intruders

These deduction systems all preserve weakening, proved as `dyder_wk`, `dyder_rt_wk` and `dyder_norm_wk`, which we use in many of the proofs described subsequently.

A.2 Invertibility and cut-admissibility

We now describe the Isabelle proof of cut-admissibility for this system.

First we proved invertibility of the right rules p_R and e_R, in the following form.

Lemma 20. *(a) (dy_pr_inv') If $\Gamma \vdash \langle M, N\rangle$ then $\Gamma \vdash M$ and $\Gamma \vdash N$*
(b) (dy_er_inv') If $\Gamma \vdash \{M\}_K$ and $\Gamma \vdash K$ then $\Gamma \vdash M$

```
val dy_pr_inv' = "(?oth, Mpair ?M ?N) : dyder ==> (?oth, ?M) : dyder & (?oth, ?N) : dyder"

val dy_er_inv' = "[| (?oth, Enc ?M ?K) : dyder; (?oth, ?K) : dyder |] ==> (?oth, ?M) : dyder"
```

Then we proved cut-admissibility in the following form:

Lemma 21. *If $\Gamma \vdash M_i$ for all $i \in I$, and $\Delta' \vdash R$, where $\Delta' \subseteq \Delta \cup \{M_i \,|\, i \in I\}$, then $\Gamma \cup \Delta \vdash R$*

```
dyder_ca = "[| (?Del', ?R) : dyder; ?Del' <= ?Del Un ?MNs;
      ALL MNi:?MNs. (?Gam, MNi) : dyder |] ==> (?Del Un ?Gam, ?R) : dyder"
```

Note how we also used a version with multiple cut-formulae, and stronger inductive hypothesis, whose assumption is an inequality ($\Delta' \subseteq \ldots$); this was much easier to prove in Isabelle than if the "$\subseteq$" were "$=$" instead.

This result was proved by induction on the derivation of $\Delta' \vdash R$. It is also possible to prove a similar cut-admissibility result (with a single cut-formula M) by induction on the derivation of $\Gamma \vdash M$: this would require first proving invertibility results for counterparts of the left rules with the principal formula omitted from the main premise.

Note that these proofs of cut-admissibility do not require an induction on the size of the cut-formula, because the none of the rules moves a formula from the right to the left (as happens with implication or negation connectives).

A.3 Existence of normal derivations

First we present a proof method which is different from that used in §4.

We define a relation $\longrightarrow$ used to transform a set of messages Γ into another set Γ', by adding components of messages in Γ, such that for each message M, $\Gamma \vdash M$ is derivable iff $\Gamma' \vdash M$ is. The idea is that if $\Gamma \vdash M$ is derivable then Γ can be transformed (in multiple steps) to Γ' ($\Gamma \longrightarrow^* \Gamma'$) such that $\Gamma' \Vdash_{\mathcal{R}} M$.

Definition 13 (`include_subs`). *We define a relation $\longrightarrow$ on message sets, where*

$$\begin{aligned} \Gamma, \langle M, N\rangle &\longrightarrow \Gamma, \langle M, N\rangle, M, N \\ \Gamma, \{M\}_K &\longrightarrow \Gamma, \{M\}_K, M, K && \text{if } \Gamma, \{M\}_K \Vdash_{\mathcal{R}} K \end{aligned}$$

with in all cases the additional requirement for $\Gamma \longrightarrow \Gamma'$, that $\Gamma \neq \Gamma'$.

We call the relation "reduction" because it adds smaller messages to Γ, even though it actually enlarges Γ.

Lemma 22. *If $\Gamma \longrightarrow \Gamma'$ then*

(a) (`is_norm`) if $\Gamma' \vdash M$ has a normal derivation then so does $\Gamma \vdash M$, and
(b) if $\Gamma \vdash M$ is derivable then so is $\Gamma' \vdash M$.

Proof. If $\Gamma' \vdash M$ has a normal derivation then applying the left rule corresponding to the relevant clause of Definition 13 gives a normal derivation of $\Gamma \vdash M$. As written, the second part depends only on weakening for $\vdash$. (In fact it is also an equivalence, by a similar argument to (a)). □

Lemma 23 (`nonr_inc`). *If $\Gamma \vdash M$ is derivable but $\Gamma \nVdash_{\mathcal{R}} M$ then Γ is $\longrightarrow$-reducible.*

Proof. The proof is by induction on the cut-free derivation of $\Gamma \vdash M$ in $\mathcal{S}$, so we look at the last rule used, and assume the result holds for all the premises of that rule. We assume a derivation with no trivial steps (where a trivial step is one where one premise is the same as the conclusion).

If the last rule used is a right rule, deriving $\Gamma \vdash f(M, N)$ from $\Gamma \vdash M$ and $\Gamma \vdash N$, then we cannot have that both $\Gamma \Vdash_{\mathcal{R}} M$ and $\Gamma \Vdash_{\mathcal{R}} N$, for then we would have $\Gamma \Vdash_{\mathcal{R}} f(M, N)$, contrary to our assumption. Therefore either $\Gamma \nVdash_{\mathcal{R}} M$ or $\Gamma \nVdash_{\mathcal{R}} N$ and so, by the induction hypothesis, Γ is reducible.

If the last rule used is the *id* rule, this gives $\Gamma \Vdash_{\mathcal{R}} M$, a contradiction.

If the last rule used is p_L, then it follows by definition that Γ is reducible.

If the last rule used is e_L, observe that the left-hand premise is $\Gamma \vdash K$ for some K. Now, if in fact $\Gamma \Vdash_{\mathcal{R}} K$ then one of the clauses defining $\longrightarrow$ applies, and so Γ is reducible. On the other hand, if $\Gamma \nVdash_{\mathcal{R}} K$ then the inductive hypothesis gives that Γ is reducible, as required. □

```
nonr_inc = "[| (?S, ?M) : dyder; (?S, ?M) ~: dyder_rt |] ==> EX S'. (S', ?S) : include_subs"
```

Assuming we work only with finite sets of messages, it is easy to see that $\longrightarrow$ is well-founded, since then only messages that may be added to Γ are sub-messages of messages originally in Γ.[5]

Lemma 24 (`wfp_is`). *If Γ is finite, then Γ is in the well-founded part of $\longrightarrow$.*

Lemma 25 (`finite_dy_norm`). *If Γ is finite then if $\Gamma \vdash M$ is derivable then it has a normal derivation.*

Proof. Since Γ is finite, by Lemma 24, Γ is in the well-founded part of $\longrightarrow$. Then we can find Γ' such that $\Gamma \longrightarrow^* \Gamma'$ and Γ' is not further reducible. Then by Lemma 22, $\Gamma' \vdash M$ is derivable. But as Γ' is not further reducible, by Lemma 23 it follows that $\Gamma' \Vdash_{\mathcal{R}} M$ and so $\Gamma' \vdash M$ has a normal derivation. Then, by Lemma 22, $\Gamma \vdash M$ has a normal derivation, as required. $\square$

Finally we use the fact that a derivation can only use finitely many messages in Γ to remove the finiteness condition from Lemma 25.

Lemma 26 (`dyder_finite_char`). *If $\Gamma \vdash M$ is derivable then $\Gamma' \vdash M$ is derivable for some finite $\Gamma' \subseteq \Gamma$.*

Proposition 8 (`dyder_norm`). *$\Gamma \vdash M$ is derivable then it has a normal derivation.*

Proof. By Lemma 26 let $\Gamma' \vdash M$ be derivable for some finite $\Gamma' \subseteq \Gamma$. Then by Lemma 25 $\Gamma' \vdash M$ has a normal derivation, and so, by weakening for normal derivations, $\Gamma \vdash M$ has a normal derivation. $\square$

```
wfp_is = "finite ?S ==> ?S : wfp include_subs"

finite_dy_norm = "[| finite ?S; (?S, ?M) : dyder |] ==> (?S, ?M) : dyder_norm"

dyder_finite_char = "(?G, ?MN) : dyder ==> EX G'<=?G. finite G' & (G', ?MN) : dyder"

dy_norm = "?x : dyder ==> ?x : dyder_norm"
```

A.4 Another proof of existence of normal derivations

We now present a proof of this result based on the proof of Proposition 4 in §4.

To express Lemma 9 we define a derivation system like that of Figure 5, except that the left-hand premise of the e_L rule is $\Gamma, \{M\}_K \vdash K$, not $\Gamma, \{M\}_K \Vdash_{\mathcal{R}} K$. We call a derivation using these rules a "semi-normal" derivation, $(\Sigma, M) \in$ `dyder_sn`. We proved that this system also has the weakening property.

Whereas Lemma 9 could be simply expressed by saying that a right rule can be permuted upwards until all right rules are above all left rules, there are in fact two separate inductive arguments involved (as the proof of Lemma 9). In the proof of Lemma 9 these are inductions on the number of "offending" right rules, and induction on the height of the derivation Π. Since, in our Isabelle formalisation, we do not explicitly model a derivation as an object whose height, or number of offending rules, can be defined, we need a slightly different proof.[6]. Each of the following lemmas contains an induction, which may be thought of as being in place of the two inductions mentioned above.

Lemma 27 (`pr_sn_adm, er_sn_adm`). *The right rules p_R and e_R are admissible for semi-normal derivations.*

Proof. That is, if there are semi-normal derivations of $\Sigma \vdash M_a$ and $\Sigma \vdash M_b$, then we need to show that there is a semi-normal derivation of $\Sigma \vdash \langle M, \rangle\, aM_b$. In fact we needed to formulate the result as: if there are semi-normal derivations of $\Sigma_a \vdash M_a$ and $\Sigma_b \vdash M_b$, then there is a semi-normal derivation of $\Sigma \vdash \langle M, \rangle\, aM_b$ (which is equivalent, by weakening for `dyder_sn`). This result is proved by induction on the fact of derivation of, firstly, $\Sigma_a \vdash M_a$, and then, secondly, $\Sigma_b \vdash M_b$. $\square$

[5] In a similar Isabelle proof, let $\longrightarrow$ remove the principal formula, then well-foundedness can be based on Γ getting smaller.

[6] This issue is described in more detail in Dawson & Goré, Generic Methods for Formalising Sequent Calculi Applied to Provability Logic, in preparation

Lemma 28 (`dy_sn`)**.** *If $\Gamma \vdash M$ is derivable then it has a semi-normal derivation.*

Proof. This proof (which is now quite simple) is by induction on the fact of derivation of $\Gamma \vdash M$. ☐

To now prove the existence of a normal derivation, we in fact use a counterpart of Lemma 27 for normal derivations, with indentical proof. Thus we do not actually use Lemmas 27 or 28 as such.

Lemma 29 (`pr_norm_adm, er_norm_adm`)**.** *The right rules p_R and e_R are admissible for normal derivations.*

We then get an admissibility result for the e_L rule (of Figure 4) for normal derivations.

Lemma 30 (`el_sn_adm`)**.** *The rule e_L is admissible for normal derivations. That is, if $\{M\}_K \in \Sigma$ and $\Sigma \vdash K$ and $\Sigma, M, K \vdash T$ have normal derivations, then $\Sigma \vdash T$ has a normal derivation.*

Proof. The proof is by induction on the fact of normal derivation of $\Sigma \vdash K$, and is the equivalent of the induction on the height of a left premise derivation in the proof of Proposition 4. ☐

Lemma 31 (`dy_norm_alt`)**.** *If $\Gamma \vdash M$ is derivable then it has a normal derivation.*

Proof. This proof is by induction on the fact of derivation of $\Gamma \vdash M$, using Lemmas 29 and 30. This induction is the counterpart of the last sentence of the proof of Proposition 4. ☐

A.5 Deducibility Constraints

We implement a deducibility constraint (Definition 7) as a triple, consisting of a boolean flag to indicate whether it is a right-deducibility constraint, a set of messages (Σ) and a message (M).

```
types
  'n ldc = "bool * 'n msg set * 'n msg"
```

We implement Definition 8, the definition of a solution to a list of deducibility constraints, by the functions `dysatldc` (for a single constraint) and `dysatldcs`(for a list of constraints). Thus `dysatldcs` θ C means that θ is a solution to C.

```
consts
  dysatldc :: "('n => 'n msg) => 'n ldc => bool"
  dysatldcs :: "('n => 'n msg) => 'n ldc list => bool"

primrec
  "dysatldc f (R, con) = (if R then subst_con f con : dyder_rt
    else subst_con f con : dyder)"

defs
  dysatldcs_def : "dysatldcs f Cs == Ball (set Cs) (dysatldc f)"
```

For Definition 9, the definition of deducibility constraint systems, we defined predicates `lhs_dycond` (for clause 1 of Definition 9), and `fnrf` (for clause 2).

More precisely, `lhs_dycond` is defined relative to a specific substitution:

```
consts
  lhs_dycond :: "'n ldc list => ('n => 'n msg) => bool"
  lhs_dycond1 :: "('n => 'n msg) => 'n msg set => 'n msg set => bool"

defs
  lhs_dycond1_def : "lhs_dycond1 f S_i S_j == ALL M : S_i.
    subst_con f (S_j - ms_ctg_fns (- UNION S_i fn_msg), M) : dyder"
```

```
primrec
  Nil : "lhs_dycond [] theta = True"
  Cons : "lhs_dycond (x # xs) theta = (lhs_dycond xs theta &
    (case x of (flag, S_j, M_j) =>
      (ALL (flag_i, S_i, M_i): set xs. lhs_dycond1 theta S_i S_j)))"
```

Here, lhs_dycond1 θ Σ_i Σ_j represents condition of clause 1 for specific θ, Σ_i and Σ_j, and lhs_dycond C θ says that this holds for θ and all constraints $\Sigma_i \Vdash^?_{(R)} M_i$ and $\Sigma_j \Vdash^?_{(R)} M_j$ $(i < j)$ in C.

Condition 2 of Definition 9 can equivalently be expressed that any variable appears in the right side of a constraint before it appears in the left side of a constraint. The Isabelle predicate fnrf says that if a variable is in the left-hand side of a constraint then it appears in the right-hand side of an earlier constraint. In fact we prove (in fnrf_ext_simp) from this that if a variable x appears in the left-hand side Σ of a constraint $\Sigma \Vdash^?_{(R)} M$ then there is an earlier constraint $\Sigma' \Vdash^?_{(R)} M'$ such that x is in M' but not in Σ'. If we take this constraint $\Sigma' \Vdash^?_{(R)} M'$ to be the first constraint containing x then we get the condition expressed in clause 2 of Definition 9.

```
consts
  fnrf :: "'n ldc list => bool"

recdef "fnrf" "measure size"
  "fnrf [] = True"
  "fnrf ((flag, S, M) # Cs) = (fnrf Cs & (ALL x : UNION S fn_msg.
      EX f' S' M'. (f', S', M') : set Cs & x : fn_msg M'))"
```

The definition of solved form, Definition 10, is given by the Isabelle predicate rsolved_form. The Isabelle predicate lsolved_form refers to the same concept without the requirement that constraints be right-constraints.

```
lsolved_form_def = "lsolved_form ?ldcs == snd ` snd ` set ?ldcs <= range Name"
rsolved_form_def = "rsolved_form ?ldcs == lsolved_form ?ldcs & Ball (set ?ldcs) fst"
```

The reduction relation is described by the Isabelle relation ndcr, defined in the file Spi_NDI_DC.thy, which partly depends on definitions in Spi_DI_DC.thy. This arrangement of files is because we also had implemented a different definition of constraint reduction, and performed proofs for a different system of deriving $\Gamma \vdash M$, where some of the work was common to the other definitions. A set of introduction rules corresponding to Definition 13 is in ndcr_eintros.

The soundness theorem, Lemma 15, is easy to prove. It does not require the conditions for C to be a deducibility constraint system.

```
ndcr_dysound = "[| (?th, ?Cs', ?Cs) : ndcr; dysatldcs ?sig ?Cs' |] ==>
  dysatldcs (comp_subst ?sig ?th) ?Cs"
```

Lemma 17 says that reduction preserves the property of being a deducibility constraint system. In Isabelle we have separate theorems for conditions 1 and 2, and the former applies "per solution".

Lemma 32. *Let* $C \overset{\theta}{\rightsquigarrow} C'$.

(a) Assume that if $\theta \circ \sigma$ *is a solution for* C, *then* lhs_dycond C $(\theta \circ \sigma)$. *Then, if* σ *is a solution for* C', *then* lhs_dycond C' σ.
(b) If fnrf C *then* fnrf C'.

```
ndcr_pres_dyc = "[| (?theta, ?Cs', ?Cs) : ndcr; ?gamma = comp_subst ?sig ?theta;
    dysatldcs ?gamma ?Cs --> lhs_dycond ?Cs ?gamma;
    dysatldcs ?sig ?Cs' |] ==> lhs_dycond ?Cs' ?sig"

ndcr_pres_fnrf = "[| (?th, ?Cs', ?Cs) : ndcr; fnrf ?Cs |] ==> fnrf ?Cs'"
```

Lemma 18, which states the well-foundedness of the reduction relation, involves the well-foundedness of the multi-set ordering. This is a well-known but non-trivial result. We prove the well-foundedness of $\rightsquigarrow$ slightly differently. We let the measure $m(C)$ of a constraint system C be the 4-tuple of the following

(a) the number of variables present
(b) the number of proper (ie, not right) constraints
(c) the sum over all proper constraints of the sum of the sizes of messages on the left-hand side of the constraint
(d) the sum of the sizes of messages on the right-hand side of constraints

If we compare measures using the lexicographic ordering of these four components, we see that if $C \overset{\theta}{\rightsquigarrow} C'$ then $m(C') < m(C)$.[7] Note that the theorem records the necessity that the left-hand sides of constraints must be finite.

```
fi_wfp_ndcr = "fst ‘ snd ‘ set ?Cs <= Finites ==> ?Cs : wfp (snd ‘ ndcr)"
```

Our proof of Lemma 19, the completeness lemma, is broken up into a number of subsidiary lemmas.

Lemma 33 (dycond_lem). *Let C' be (an initial part of) a constraint system where the right-hand sides of C' are variables, and let x be one of those variables. Let θ be a solution for C'. Let* **lhs_dycond1** *θ Σ' Σ for all $\Sigma' \Vdash^?_{(R)} M' \in C'$ (as would hold if $C'; \ldots ; \Sigma \Vdash^?_{(R)} M; \ldots$ were a constraint system). Then $(\Sigma \setminus \mathsf{V})\theta \vdash x\theta$.*

Proof. By induction on the structure of C'. It is trivial when $C' = []$, so let $C' = C''; \Sigma_i \Vdash^?_{(R)} x_i$, and we need to prove that $(\Sigma \setminus \mathsf{V})\theta \vdash x_i\theta$. Since θ is a solution for C', $\Sigma_i\theta \vdash x_i\theta$, and so, by cut-admissibility, it is enough to show that $(\Sigma \setminus \mathsf{V})\theta \vdash \Sigma_i\theta$. Now since lhs_dycond1$\theta$ Σ_i Σ we have that $\Sigma^{dv}\theta \vdash \Sigma_i\theta$, where Σ^{dv} is obtained from Σ by deleting messages containing variables not in any message in Σ_i. By cut-admissibility again therefore, it is enough to show that $(\Sigma \setminus \mathsf{V})\theta \vdash \Sigma^{dv}\theta$. Given the *id* rule, it therefore only remains to show that $(\Sigma \setminus \mathsf{V})\theta \vdash x_j\theta$ for $x_j \in \Sigma^{dv}$. Since each such x_j is the right-hand side of a constraint $\Sigma_j \Vdash^?_{(R)} x_j$ in C'', the induction hypothesis shows that $(\Sigma \setminus \mathsf{V})\theta \vdash x_j\theta$, as required. □

```
dycond_lem = "[| (snd o snd) ‘ set ?C <= range Name; fnrf ?C; dysatldcs ?theta ?C;
    ALL (flag’, S’, M’):set ?C. lhs_dycond1 ?theta S’ ?S; ?x : (snd o snd) ‘ set ?C |] ==>
    (subst_msg ?theta ‘ (?S - range Name), subst_msg ?theta ?x) : dyder"
```

We then apply this lemma to the case of a constraint system $C'; \Sigma \Vdash^?_{(R)} M$.

Lemma 34 (dycond_lem_ext). *Let $C = C'; \Sigma \Vdash^? M$ be a constraint system where the right-hand sides of C' are variables. Let θ be a solution for C. Then $(\Sigma \setminus \mathsf{V})\theta \vdash M\theta$.*

Proof. For each variable x which is a message in Σ, we have by Definition 9, condition 2, that x appears in the right-hand side of an earlier constraint $\Sigma' \Vdash^?_{(R)} M'$. Then, by the assumption, M' is equal to x, and so, by Lemma 33, $(\Sigma \setminus \mathsf{V})\theta \vdash x\theta$. Since we have $\Sigma\theta \vdash M\theta$, repeated use of the cut-admissibility lemma gives $(\Sigma \setminus \mathsf{V})\theta \vdash M\theta$. □

```
dycond_lem_ext =
  "[| (snd o snd) ‘ set ?Cs’ <= range Name; ?Cs = (?flag, ?S, ?M) # ?Cs’;
    fnrf ?Cs; dysatldcs ?theta ?Cs; lhs_dycond ?Cs ?theta |] ==>
  (subst_msg ?theta ‘ (?S - range Name), subst_msg ?theta ?M) : dyder"
```

Lemma 35 (sub_nt_dyrednre). *Let $(\Sigma \setminus \mathsf{V})\theta \vdash M\theta$, where $\Sigma \Vdash^? M$ is a constraint in the system C, and let θ be a solution for C. Then there exists a reduction $C \rightsquigarrow C'$ such that θ is a solution for C'.*

[7] Our first attempt at a proof failed because we omitted the second component of the ordering — then a reduction by rule **C3** fails if it involves a constraint whose left-hand side Σ is empty.

Proof. Consider a normal derivation of $(\Sigma\backslash\mathsf{V})\theta \vdash M\theta$. If the last rule is a right rule or *id*, then $(\Sigma\backslash\mathsf{V})\theta \Vdash_{\mathcal{R}} M\theta$ whence $\Sigma\theta \Vdash_{\mathcal{R}} M\theta$ (by weakening) and so the constraint may be changed to $\Sigma \Vdash^?_R M$, that is, a reduction by **C3**. Otherwise, the last rule is p_L or e_L, and because the principal formula is not a substituted variable, then reductions **C4** or **C5** apply to $\Sigma \setminus \mathsf{V} \Vdash^? M$ and so to $\Sigma \Vdash^? M$.[8]. □

```
sub_nt_dyrednre =
  "[| (subst_msg ?theta ‘ (?S - range Name), subst_msg ?theta ?M) : dyder;
    (False, ?S, ?M) : set ?Cs; dysatldcs ?theta ?Cs |] ==>
    EX Cs’. (Name, Cs’, ?Cs) : ndcr & dysatldcs ?theta Cs’"
```

Lemma 36 (`unrs_nfnrf_dye`)**.** *Let C be a constraint system, not in solved form, and let θ be a solution for C. Then there exists a reduction $C \overset{\sigma}{\rightsquigarrow} C'$ such that $\theta = \sigma \circ \gamma$ and γ is a solution for C'.*

Proof. Since C is not in solved form, let $\Sigma \Vdash^?_{(R)} M$ be the first constraint where M is not a variable. So, by Lemma 34, $(\Sigma \setminus \mathsf{V})\theta \vdash M\theta$. Then, if it is a proper deducibility constraint, Lemma 35 gives the required reduction. If it is a right-deducibility constraint, then since $\Sigma\theta \Vdash_{\mathcal{R}} M\theta$, then the last rule of a derivation of $\Sigma\theta \vdash M\theta$ is either a right rule or the *id* rule. If it is the *id* rule, this gives a reduction by **C1**. If it is a right rule, then, since M is not a variable, then there is a reduction of $\Sigma \Vdash^?_R M$ by **C2**. □

```
unrs_nfnrf_dye = "[| ~ rsolved_form ?Cs; dysatldcs ?theta ?Cs; fnrf ?Cs;
    lhs_dycond ?Cs ?theta; ALL (f, S, M):set ?Cs. finite S |] ==>
  EX Cs’ sig gamma.  (sig, Cs’, ?Cs) : ndcr & dysatldcs gamma Cs’ &
    ?theta = comp_subst gamma sig"
```

Finally, the completeness lemma depends on the well-foundedness of reduction and on Lemma 36.

Lemma 37 (`dyndcr_complete`)**.** *Let C be a constraint system and let θ be a solution for C. Then there exists σ such that $C \overset{\sigma}{\Longrightarrow} C'$, C' is in solved form, $\theta = \sigma \circ \gamma$, and γ is a solution for C'.*

Proof. Given constraint system C, if it is not in solved form, apply Lemma 36 to find a reduction to C'. By Lemma 32 C' is also a constraint system, and so if C' is not in solved form we can continue in this way. But by the well-foundedness of reduction this procedure terminates. □

Note that in the version we proved in Isabelle, Definition 9, condition 1 need apply only for the specific θ given.

```
dyndcr_complete = "[| dysatldcs ?theta ?Cs; fnrf ?Cs; lhs_dycond ?Cs ?theta;
            fst ‘ snd ‘ set ?Cs <= Finites |] ==>
  EX sig gamma Cs’. (sig, Cs’, ?Cs) : sig_rtc ndcr & dysatldcs gamma Cs’ &
    rsolved_form Cs’ & ?theta = comp_subst gamma sig"
```

[8] We don’t explicitly use Lemma 16, but here is where we use the reasoning behind it