Chapter 2

Equality

Decision Procedures
Last update of lecture notes: Tuesday, March 7, 2006
Last update of this chapter: Monday, February 27, 2006

2.1 The modelclass

2.1.1 Definition
Let Σ be a signature. The modelclass of equality over Σ is the modelclass

$$M_{\approx}^{\Sigma} = (\Sigma, \mathbf{A}) \,,$$

where $\mathbf{A}$ is the class of all Σ-structures.

2.1.2 Proposition
Every Σ-formula φ is $M_{\approx}^{\Sigma}$-valid if and only if it is valid.

Proof. Immediate.

2.1.3 Proposition
Every Σ-formula φ is $M_{\approx}^{\Sigma}$-satisfiable if and only if it is satisfiable.

Proof. Immediate.

2.2 Congruence closure

2.2.1 Definition
Let T be a set of terms. A congruence relation of T is a binary relation R of T satisfying the following properties:

1. R is an equivalence relation of T.
2. If $(s_i, t_i) \in R$, for $i = 1, \ldots, n$, and $f(s_1, \ldots, s_n), f(t_1, \ldots, t_n) \in T$ then $(f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)) \in R$.

2.2.2 Definition

Let T be a set of terms, and let R be a binary relation of T. The CONGRUENCE CLOSURE of R with respect to T is the unique binary relation C of T satisfying the following properties:

1. C is a congruence relation of T.

2. If R' is a congruence relation of T and $R \subseteq R'$ then $C \subseteq R'$.

2.2.3 Definition

Let T be a set of terms. A binary relation R of T is WELL-SORTED if

$$(s,t) \in R \implies s \text{ and } t \text{ have the same sort}, \qquad \text{for all } s,t \in T\,.$$

2.2.4 Proposition

Let T be a set of terms, and let R be a binary relation of T. Assume that R is well-sorted. Then the congruence closure C of R with respect to T is well-sorted.

PROOF. Let

$$R' = C \setminus \{(s,t) \in C \mid s \text{ and } t \text{ do not have the same sort}\}\,.$$

By construction, R' is a well-sorted congruence relation of T such that $R \subseteq R' \subseteq C$. But then, $C \subseteq R'$, which implies $R' = C$. It follows that C is well-sorted.

2.2.5 Algorithm (IS-SATISFIABLE-EQUALITY)

Input: A conjunction Γ of Σ-literals

Output: `satisfiable` if Γ is satisfiable; `unsatisfiable` otherwise

1: **function** IS-SATISFIABLE-EQUALITY(Γ)
2: $\quad T \leftarrow$ the set of all terms occurring in Γ
3: $\quad R \leftarrow \{(s,t) \in T \times T \mid \text{the literal } s \approx t \text{ is in } \Gamma\}$
4: $\quad C \leftarrow$ the congruence closure of R with respect to T
5: $\quad$ **if** there exist a literal $s \not\approx t$ in Γ such that $(s,t) \in C$ **then**
6: $\quad\quad$ **return** `unsatisfiable`
7: $\quad$ **else if** there exists literals $p(s_1,\ldots,s_n)$ and $\neg p(t_1,\ldots,t_n)$ in Γ such that $(s_i,t_i) \in C$, for $i = 1,\ldots,n$ **then**
8: $\quad\quad$ **return** `unsatisfiable`
9: $\quad$ **else**
10: $\quad\quad$ **return** `satisfiable`
11: $\quad$ **end if**
12: **end function**

2.2.6 Proposition

If Algorithm IS-SATISFIABLE-EQUALITY *terminates at line 10, returning* `satisfiable`*, then Γ is satisfiable.*

PROOF. Assume that Algorithm IS-SATISFIABLE-EQUALITY terminates at line 10, returning `satisfiable`, We construct a Σ-interpretation $\mathcal{A}$ over $vars(\Gamma)$ as follows.

For each sort $\sigma \in \Sigma^{\mathrm{S}}$ such that $T_\sigma = \varnothing$, fix some arbitrary object a_σ. Moreover, for each sort $\sigma \in \Sigma^{\mathrm{S}}$ such that $T_\sigma \neq \varnothing$, fix a term $t_\sigma \in T_\sigma$.

Then, for each $\sigma \in \Sigma_{\mathrm{S}}$, we let

$$A_\sigma = \begin{cases} T_\sigma / C\,, & \text{if } T_\sigma \neq \varnothing\,,\\ \{a_\sigma\}\,, & \text{otherwise}\,. \end{cases}$$

Moreover, we let

- for variables $x \in \mathit{vars}(\Gamma)$:

$$x^{\mathcal{A}} = [x]_C$$

- for constant symbols $c \in \Sigma^{\mathrm{C}}$:

$$c^{\mathcal{A}} = \begin{cases} [c]_C\,, & \text{if } c \in T\,,\\ [t_\sigma]_C\,, & \text{otherwise}\,. \end{cases}$$

- for function symbols $f \in \Sigma^{\mathrm{F}}$:

$$f^{\mathcal{A}}([t_1]_C, \ldots, [t_n]_C) = \begin{cases} [f(s_1, \ldots, s_n)]_C\,, & \begin{array}{l}\text{if } f(s_1, \ldots, s_n) \in T \text{ and}\\ (s_i, t_i) \in C, \text{ for all } i = 1, \ldots, n\end{array}\,,\\ [t_\sigma]_C\,, & \text{otherwise}\,. \end{cases}$$

- for predicate symbols $p \in \Sigma^{\mathrm{P}}$:

$$([t_1]_C, \ldots, [t_n]_C) \in p^{\mathcal{A}} \quad\Longleftrightarrow\quad \begin{array}{l}\text{a literal } p(s_1, \ldots, s_n) \text{ is in } \Gamma \text{ and}\\ (s_i, t_i) \in C, \text{ for all } i = 1, \ldots, n\ .\end{array}$$

By structural induction, one can verify that

$$t^{\mathcal{A}} = [t]_C\,, \qquad\qquad \text{for all } t \in T\,.$$

Next, we prove that $\mathcal{A}$ satisfies all literals in Γ.

- *Literals of the form $s \approx t$.*

  Let the literals $s \approx t$ be in Γ. Then $(s,t) \in R$ which implies $(s,t) \in C$. Thus, $s^{\mathcal{A}} = [s]_C = [t]_C = t^{\mathcal{A}}$.

- *Literals of the form $s \not\approx t$.*

  Suppose, by contradiction, that $s^{\mathcal{A}} = t^{\mathcal{A}}$. It follows that $[s]_C = [t]_C$. But then, the algorithm would have ended at line 6 returning `unsatisfiable`.

- *Literals of the form $p(t_1, \ldots, t_n)$.*

  By construction, $([t_1]_C, \ldots, [t_n]_C) \in p^{\mathcal{A}}$, which implies that $(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}) \in p^{\mathcal{A}}$.

- *Literals of the form* $\neg p(t_1, \ldots, t_n)$.

  Suppose, by contradiction, that $(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}) \in p^{\mathcal{A}}$. It follows that $([t_1]_C, \ldots, [t_n]_C) \in p^{\mathcal{A}}$. Therefore, there exists a literal $p(s_1, \ldots, s_n)$ in Γ such that $(s_i, t_i) \in C$, for all $i = 1, \ldots, n$. But then, the algorithm would have ended at line 8 returning `unsatisfiable`.

2.2.7 Proposition
If Algorithm IS-SATISFIABLE-EQUALITY *terminates at either line 6 or line 8, returning* `unsatisfiable`*, then* Γ *is unsatisfiable.*

PROOF. Assume that algorithm IS-SATISFIABLE-EQUALITY returns `unsatisfiable`. By contradiction, assume that Γ is satisfiable. Then there exists a Σ-interpretation $\mathcal{A}$ over $vars(\Gamma)$ such that $\mathcal{A} \models \Gamma$.

Let R' be the binary relation of T defined by

$$(s, t) \in R' \iff s^{\mathcal{A}} = t^{\mathcal{A}} \,.$$

By construction, R' is a congruence relation of T. Moreover, $R \subseteq R'$. Therefore, it follows $C \subseteq R'$.

If the algorithm ended at line 6, then there exists a literal $s \not\approx t$ in Γ such that $(s, t) \in C$. But then $(s, t) \in R'$ which implies $s^{\mathcal{A}} = t^{\mathcal{A}}$, contradicting $\mathcal{A} \models \Gamma$.

If instead the algorithm ended at line 8, then there exist literals $p(s_1, \ldots, s_n)$ and $\neg p(t_1, \ldots, t_n)$ such that $(s_i, t_i) \in C$, for all $i = 1, \ldots, n$. But then $(s_i, t_i) \in R'$, for all $i = 1, \ldots, n$. It follows that $s_i^{\mathcal{A}} = t_i^{\mathcal{A}}$, for all $i = 1, \ldots, n$, which contradicts $\mathcal{A} \models \Gamma$.

2.2.8 Proposition
Algorithm IS-SATISFIABLE-EQUALITY *is correct.*

PROOF. Termination is obvious. Partial correctness follows by Propositions 2.2.6 and 2.2.7.

2.3 Nelson-Oppen

2.3.1 Algorithm (NELSON-OPPEN-CONGRUENCE-CLOSURE**)**
Input: A finite set T of terms and a binary relation R of T
Output: The congruence closure C of R with respect to T.

```
function NELSON-OPPEN-CONGRUENCE-CLOSURE(R, T)
    C ← {(t, t) | t ∈ T}
    for all (s, t) ∈ R do
        MERGE(s, t)
    end for
    return C
end function
```

```
8: procedure MERGE(s, t)
9:     if (s, t) ∉ C then
10:        P ← PREDS(s)
11:        Q ← PREDS(t)
12:        UNION(s, t)
13:        for all (u, v) ∈ P × Q do
14:            if (u, v) ∉ C and CONGRUENT(u, v) then
15:                MERGE(u, v)
16:            end if
17:        end for
18:    end if
19: end procedure

20: procedure UNION(s, t)
21:    C ← (C ∪ {(s, t), (t, s)})*
22: end procedure

23: function PREDS(t)
24:    return {u ∈ T | u ≡ f(..., t′, ...) and (t, t′) ∈ C}
25: end function

26: function CONGRUENT(u, v)
27:    if u ≡ f(s_1, ..., s_n), v ≡ f(t_1, ..., t_n), and (s_i, t_i) ∈ C, for all i =
       1, ..., n then
28:        return true
29:    else
30:        return false
31:    end if
32: end function
```

2.3.2 Proposition
Algorithm NELSON-OPPEN-CONGRUENCE-CLOSURE *terminates.*

PROOF. It suffices to prove that the number of calls to UNION is finite.

Note that C is initialized at line 2, and modified only by the procedure UNION at line 21. Moreover, each call to UNION strictly increases the value of $|C|$. Since this value cannot be greater than $|T \times T|$, it follows that UNION can be called only a finite number of times.

2.3.3 Proposition
In Algorithm NELSON-OPPEN-CONGRUENCE-CLOSURE, *C is always an equivalence relation of T.*

PROOF. Let

$$C_0, C_1, \ldots, C_k, \ldots, C_m\,,$$

be the values taken by C during the execution of the algorithm.

Since C is initialized at line 2 and modified at line 21, we have:

- $C_0 = \{(t,t) \mid t \in T\}$.
- C_m is the value returned by the function NELSON-OPPEN-CONGRUENCE-CLOSURE.
- For $0 \le k < n$, C_k is the value of C just before the k-th call to the procedure UNION, whereas C_{k+1} is the value of C just after that call.
- For $0 \le k < n$, we have

$$C_{k+1} = (C_k \cup \{(s,t),(t,s)\})^*\,, \qquad \text{for some terms } s,t \in T\,.$$

We want to show that C_k is an equivalence relation, for all k. We can do this by induction on k.

For the base step, C_0 is clearly an equivalence relation. For the induction step, suppose that C_k is an equivalence. Then clearly $C_{k+1} = (C_k \cup \{(s,t),(t,s)\})^*$ is also an equivalence relation.

2.3.4 Proposition
At the end of the execution of NELSON-OPPEN-CONGRUENCE-CLOSURE, *we have* $R \subseteq C$.

PROOF. Let

$$C_0, C_1, \ldots, C_k, \ldots, C_m\,,$$

be the values taken by C during the execution of the algorithm. We want to show that $R \subseteq C_m$.

Clearly, $C_0 \subseteq C_1 \subseteq C_2 \subseteq \cdots \subseteq C_m$.

Next, assume that $(s,t) \in R$. Then we eventually call MERGE(s,t) at line 4. At this point, if $(s,t) \in C_k$ then $(s,t) \in C_m$. Otherwise, we eventually call UNION(s,t) at line 12, which guarantees that $(s,t) \in C_m$.

2.3.5 Proposition
At the end of the execution of NELSON-OPPEN-CONGRUENCE-CLOSURE, *C is a congruence relation of T.*

PROOF. Let

$$C_0, C_1, \ldots, C_k, \ldots, C_m\,,$$

be the values taken by C during the execution of the algorithm. We want to show that C_m is a congruence relation of T.

By Proposition 2.3.3, C_m is an equivalence relation of T.

Next, assume that $(s_i, t_i) \in C_m$, for $i = 1, \ldots, n$, and that $f(s_1, \ldots, s_n)$, $f(t_1, \ldots, t_n) \in T$. Let $s \equiv f(s_1, \ldots, s_n)$ and $t \equiv f(t_1, \ldots, t_n)$.

If $s_i \equiv t_i$, for $i = 1, \ldots, n$ then $s \equiv t$, which implies $(s,t) \in C_0 \subseteq C_m$. Otherwise, there exists an index k such that after the k-th call to UNION we have

$(s_i, t_i) \in C_{k+1}$, for all $i = 1, \ldots, n$, but before that call we have $(s_j, t_j) \notin C_k$, for some $1 \leq j \leq n$. We have

$$C_{k+1} = (C_k \cup \{(u, v), (v, u)\})^*, \qquad \text{for some terms } u, v \in T.$$

Moreover, without loss of generality we can assume that $(u, s_j) \in C_k$ and $(v, t_j) \in C_k$. But then, just before the call to UNION(u, v) at line 12, we have $f(s_1, \ldots, s_n) \in$ PREDS(u) and $f(t_1, \ldots, t_n) \in$ PREDS(v). Moreover, after the call to UNION(u, v) at line 12, we have that CONGRUENT(s, t) returns *true*. Thus, we eventually call MERGE(s, t) at line 15, which guarantees that $(s, t) \in C_n$.

2.3.6 Proposition
Let R' be any congruence relation of T such that $R \subseteq R'$. Then, at the end of the execution of NELSON-OPPEN-CONGRUENCE-CLOSURE*, we have $C \subseteq R'$.*

Proof. Let

$$C_0, C_1, \ldots, C_k, \ldots, C_m,$$

be the values taken by C during the execution of the algorithm.

We prove that $C_k \subseteq R'$, for all k. We proceed by induction on k. For the base step, we clearly have $C_0 \subseteq R'$.

For the induction step, let $C_{k+1} = (C_k \cup \{(s, t), (t, s)\})^*$. Then we called UNION$(s, t)$ because either $(s, t) \in R$ or CONGRUENT(s, t) returned *true*. We prove that in both cases we must have $C_{k+1} \subseteq R'$.

Assume first that $(s, t) \in R$, and let $(u, v) \in C_{k+1}$. If $(u, v) \in C_k$ then by the induction hypothesis $(u, v) \in R'$. Otherwise, without loss of generality, we have $(u, s) \in C_k$ and $(v, t) \in C_k$. By the induction hypothesis, it follows that $(u, s) \in R'$ and $(v, t) \in R'$. Moreover, we have $(s, t) \in R'$ because $R \subseteq R'$. Since R' is an equivalence relation, we have $(u, v) \in R'$.

Finally, assume that CONGRUENT(s, t) returned *true*, and let $(u, v) \in C_{k+1}$. If $(u, v) \in C_k$ then by the induction hypothesis $(u, v) \in R'$. Otherwise, without loss of generality, we have $(u, s) \in C_k$ and $(v, t) \in C_k$. By induction the induction hypothesis, it follows $(u, s) \in R'$ and $(v, t) \in R'$. Next, let $s \equiv f(s_1, \ldots, s_n)$ and $t \equiv f(t_1, \ldots, t_n)$. Since CONGRUENT$(s, t)$ returned *true*, it follows that $(s_i, t_i) \in C_k$, for all $i = 1, \ldots, n$. By the induction hypothesis, we have $(s_i, t_i) \in R'$, for all $i = 1, \ldots, n$. Since R' is a congruence relation of T, it follows that $(s, t) \in R'$. Since R' is an equivalence relation, we have $(u, v) \in R'$.

2.3.7 Proposition
Algorithm NELSON-OPPEN-CONGRUENCE-CLOSURE *is correct.*

Proof. Termination follows by Proposition 2.3.2. Partial correctness follows by Propositions 2.3.4, 2.3.5, and 2.3.6.