# Chapter 6

# Decidable Logics

This chapter is about decidable logics. There are many decidable fragments of first-order logic, some of them are discussed in Chapter 3 and Chapter 5. Here I discuss logics that are typically not representable in first-order logic, e.g., linear integer arithmetic, Section 6.2, or logics where specialized decision procedures exist, beyond the general procedures discussed in previous chapters, e.g., equational reasoning on ground terms by congruence closure, Section 6.1, that can also be solved by Knuth-Bendix completion, Chapter 4.

## 6.1 Congruence Closure

In general, satisfiability of first-order formulas with respect to equality is undecidable. Even the word problem for conjunctions of equations is undecidable. However, I will show that satisfiability is decidable for *ground* first-order formulas.

It suffices to consider conjunctions of literals. Arbitrary ground formulas can be converted into DNF, potentially at the price of an exponential blow up. A formula in DNF is satisfiable if and only if one of its conjunctions is satisfiable. So it is sufficient to consider a conjunction of ground literals, e.g., a conjunction of ground equations.

Note that the problem can be written in several ways. An equational clause

$$\forall \vec{x}\,(t_1 \approx s_1 \vee \ldots \vee t_n \approx s_n \vee l_1 \not\approx r_1 \vee \ldots \vee l_k \not\approx r_k)$$

is valid iff

$$\exists \vec{x}\,(t_1 \not\approx s_1 \wedge \ldots \wedge t_n \not\approx s_n \wedge l_1 \approx r_1 \wedge \ldots \wedge l_k \approx r_k)$$

is unsatisfiable iff the Skolemized (ground!) formula

$$(t_1 \not\approx s_1 \wedge \ldots \wedge t_n \not\approx s_n \wedge l_1 \approx r_1 \wedge \ldots \wedge l_k \approx r_k)\{\vec{x} \mapsto \vec{c}\}$$

is unsatisfiable iff the formula

$$(t_1 \approx s_1 \vee \ldots \vee t_n \approx s_n \vee l_1 \not\approx r_1 \vee \ldots \vee l_k \not\approx r_k)\{\vec{x} \mapsto \vec{c}\}$$

is valid.

Please note validity of these transformations do depends on the shape of the (starting) formula. Validity is no preserved in case of a quantifier alternation or an existentially quantified formula, in general, or the eventual formula must not be ground. There is no way to transform a first-order (equational) formula into a ground formula preserving validity, in general.

The theory is also known as *EUF* (equality with uninterpreted function symbols) and one of the standard theories considered in *SMT* (Satisfiability Modulo Theories). The decision procedure discussed here is based on congruence closure.

The goal of the procedure is to check (un-)satisfiability of a ground conjunction

$$s_1 \not\approx t_1 \wedge \ldots \wedge s_k \not\approx t_k \wedge l_1 \approx r_1 \wedge \ldots \wedge l_n \approx r_n$$

The main idea is to transform the equations $E = \{l_1 \approx r_1, \ldots, l_n \approx r_n\}$ into an equivalent convergent TRS $R$ and check whether $s_i{\downarrow_R} = t_i{\downarrow_R}$. If $s_i{\downarrow_R} = t_i{\downarrow_R}$ for some $i$ then because $s_i{\downarrow_R} = t_i{\downarrow_R}$ iff $s_i \leftrightarrow_E^* t_i$ iff $E \models s_i \approx t_i$ (see Chapter 4) the overall conjunction is unsatisfiable. If $s_i{\downarrow_R} = t_i{\downarrow_R}$ for no $i$, i.e., $s_i{\downarrow_R} \neq t_i{\downarrow_R}$ for all $i$ then $\mathcal{I}_E$ is a model of both the equations $l_i \approx r_i$, and the inequations $s_j \not\approx t_j$. Hence the overall conjunction is satisfiable.

Knuth-Bendix completion, Chapter 4, can be used to convert $E$ into an equivalent convergent TRS $R$. If done properly, Knuth-Bendix completion always terminates for ground inputs. However, for the ground case, the procedure can be further optimized.

The first step is to introduce additional "names", i.e, extra constants for all non-constant subterms. This implements implicitly sharing among subterms.

Let $E = [l_1 \approx r_1, \ldots, l_n \approx r_n]$ be a sequence of equations interpreted as a conjunction.

**Flattening** $E \Rightarrow_{\mathrm{CCF}} E[c]_{p_1,\ldots,p_k}[f(t_1, \ldots, t_n) \approx c]$

provided $E|_{p_1,\ldots,p_k} = f(t_1, \ldots, t_n)$, all $t_i$ are constants, the $p_j$ are all positions in $E$ of $f(t_1, \ldots, t_n)$, $|p_l| > 2$ for some $l$, or, $p_l = m.2$ and $E|_{m.1}$ is not a constant for some $m$, and $c$ is fresh

Here I consider $E$ to be a sequence of equations in order for the positions to make sense. Note that after applying flattening to some term $f(t_1, \ldots, t_n)$ it cannot be applied a second time, because the position $p$ pointing to $f(t_1, \ldots, t_n)$ in $E[c]_{p_1,\ldots,p_k} \wedge f(t_1, \ldots, t_n) \approx c$ has size 2, i.e., $|p| = 2$.

For example, the system $E = [g(a, h(h(b))) \approx h(a)]$ is eventually replaced by $E = [h(b) \approx c_3, h(c_3) \approx c_4, h(a) \approx c_5, g(a, c_4) \approx c_5]$.

As a result: only two kinds of equations left. Term equations: $f(c_{i_1}, \ldots, c_{i_n}) \approx c_{i_0}$ and constant equations: $c_i \approx c_j$. This can be further explored in an implementation by specific data structures. In particular, a union-find data structure efficiently represents the equivalence classes encoded by the constant equations (rules).

The congruence closure algorithm is presented as a set of abstract rewrite rules operating on a pair of equations $E$ and a set of rules $R$, $(E; R)$, similar to Knuth-Bendix completion, Section 4.4.

$(E_0; R_0) \Rightarrow_{\mathrm{CC}} (E_1; R_1) \Rightarrow_{\mathrm{CC}} (E_2; R_2) \Rightarrow_{\mathrm{CC}} \ldots$

At the beginning, $E = E_0$ is a set of constant equations $c_i \approx c_j$ and $R_0$ is the set of term equations $f(c_{i_1}, \ldots, c_{i_n}) \approx c_{i_0}$ oriented from left-to-right. At termination, $E$ is empty and $R$ contains the result. By exhaustive application of Flattening any conjunction of equations can be transformed into this form, preserving satisfiability. Recall that the atom $s \mathrel{\dot{\approx}} t$ denotes either $s \approx t$ or $t \approx s$.

**Simplify** $(E \uplus \{c \mathrel{\dot{\approx}} c'\}; R \uplus \{c \to c''\}) \Rightarrow_{\mathrm{CC}} (E \cup \{c'' \mathrel{\dot{\approx}} c'\}; R \cup \{c \to c''\})$

**Delete** $(E \uplus \{c \approx c\}; R) \Rightarrow_{\mathrm{CC}} (E; R)$

**Orient** $(E \uplus \{c \mathrel{\dot{\approx}} c'\}; R) \Rightarrow_{\mathrm{CC}} (E; R \cup \{c \to c'\})$
if $c \succ c'$

**Deduce** $(E; R \uplus \{t \to c,\, t \to c'\}) \Rightarrow_{\mathrm{CC}} (E \cup \{c \approx c'\}; R \cup \{t \to c\})$

**Collapse** $(E; R \uplus \{t[c]_p \to c',\, c \to c''\}) \Rightarrow_{\mathrm{CC}} (E; R \cup \{t[c'']_p \to c',\, c \to c''\})$
$p \neq \epsilon$

For rule Deduce, $t$ is either a term of the form $f(c_1, \ldots, c_n)$ or a constant $c_i$. For rule Collapse, $t$ is always of the form $f(c_1, \ldots, c_n)$ For ground rewrite rules, critical pair computation does not involve substitution. Therefore, every critical pair computation can be replaced by a simplification, either using Deduce or Collapse.

The inference rules are usually applied according to the following strategy: Simplify, Delete and Orient are preferred over Deduce and Collapse. Then if Collapse becomes applicable, it is applied exhaustively.

Instead of fixing the ordering $\succ$ in advance, it is preferable to define it on the fly during the algorithm: if an equation $c \approx c'$ between two constants is oriented, a good heuristic is to make that constant symbol larger that occurs less often in $R$, hence producing afterwards fewer Collapse steps.

The average runtime of the algorithm is $O(m \log m)$, where $m$ is the number of edges in the graph representation of the initial constant and term equations.

The inference rules are sound in the usual sense. The conclusions are entailed by the premises, so every model of the premises is a model of the conclusions.

For the initial flattening rule , however, only a weaker result holds. The models of the original equations have to be extended by interpretations for the

freshly introduced constants to obtain models of the flattened equations. The result is a new algebra with the same universe as the old one, with the same interpretations for old functions and predicate symbols, but with appropriately chosen interpretations for the new constants.

Consequently, the relations $\approx_E$ and $\approx_R$ for the original $E$ and the final $R$ are not the same. On the other hand, the model extension preserves the universe and the interpretations for old symbols. Therefore, if $s$ and $t$ are terms over the old symbols, we have $s \approx_E t$ iff $s \approx_R t$. This is sufficient for our purposes: The terms $s_i$ and $t_i$ that we want to normalize using $R$ do not contain new symbols.

### 6.1.1 History

Congruence closure algorithms have been published, among others, by Shostak (1978). by Nelson and Oppen (1980), and by Downey, Sethi and Tarjan (1980).

Kapur (1997) showed that Shostak's algorithm can be described as a completion procedure.

Bachmair and Tiwari (2000) did this also for the Nelson/Oppen and the Downey/Sethi/Tarjan algorithm.

The algorithm presented here is the Downey/Sethi/Tarjan algorithm in the presentation of Bachmair and Tiwari.

## 6.2 Linear Arithmetic

There are several ways of introducing linear arithmetic and in particular its syntax. I start with a syntax that already contains $-$, $\leq$, $<$, $\geq$, $\not\approx$ and $\mathbb{Q}$. All these functions and relations are indeed expressible by first-order formulas over 0, 1, $\approx$, and $>$. For the semantics there are two approaches. Either providing axioms, i.e., closed formulas, for the above symbols and then considering all algebras satisfying the axioms, or fixing one particular algebra or a class of algebras. For this chapter I start with a rich syntax and a semantics based on a fixed algebra.

**Definition 6.2.1** (LA Syntax)**.** The syntax of LA is

$$\Sigma_{\mathrm{LA}} = (\{\mathrm{LA}\}, \{0, 1, +, -\} \cup \mathbb{Q}, \{\leq, <, \not\approx, >, \geq\})$$

where $-$ is unitary and all other symbols have the usual arities.

Terms and formulas over $\Sigma_{\mathrm{LA}}$ are built in the classical free first-order way, see Section 3.1. All first-order notions, i.e., terms, atoms, equations, literals, clauses, etc. carry over to LA formulas. The atoms and terms built over the LA signature are written in their standard infix notation, i.e., I write $3 + 5$ instead of $+(3, 5)$. Note that the signature does not contain multiplication. A term $3x$ is just an abbreviation for a term $x + x + x$. For the isolation of variables in inequations, we will temporarily introduce also multiplication of a variable with

# Chapter 7

# Propositional Logic Modulo Theories

In Chapter 6 I have studied a number of decision procedures for conjunctions of literals of some specific first-order theory or fragment. In this chapter the decision procedures are extended in two different ways. Firstly, by considering conjunctions of literals over several first-order theories. The respective procedure is the Nelson-Oppen combination procedure for theories [66], Section 7.1. Secondly, I lift the procedure from conjunctions of literals to arbitrary boolean combinations of literals. The respective procedure is CDCL(T), Section 7.2.

## 7.1 Nelson-Oppen Combination

Here I discuss a basic variant of the Nelson-Oppen [66] (NO) combination procedure for two theories $\mathcal{T}_1$ and $\mathcal{T}_2$ (see Definition 3.17.1) over two respective signatures $\Sigma_1$ and $\Sigma_2$ that do not share any function, constant, or predicate symbols, but may share sorts. The idea of the procedure is to reduce satisfiability of a quantifier-free formula over $\Sigma_1 \cup \Sigma_2$ to satisfiability of two separate formulas over $\Sigma_1$ and $\Sigma_2$, respectively.

The underlying semantics is that a quantifier-free formula $\phi$ over $\Sigma_1 \cup \Sigma_2$ is satisfiable if there exists a $\Sigma_1 \cup \Sigma_2$ algebra $\mathcal{A}$ such that $\mathcal{A}(\beta) \models \phi$ for some assignment $\beta$, and $\mathcal{A}|_{\Sigma_1}$ is isomorphic to a model in $\mathcal{T}_1$ and $\mathcal{A}|_{\Sigma_2}$ is isomorphic to a model in $\mathcal{T}_2$. Here $\mathcal{A}|_{\Sigma}$ denotes the restriction of $\mathcal{A}$ to the symbols in $\Sigma$. With appropriate restrictions, see below, the problem of testing satisfiability of $\phi$ can actually be reduced to solving finitely many separate satisfiability problems in $\Sigma_1$ and $\Sigma_2$, respectively.

Note that both theories share the equality symbol, because it is part of the first-order operator language. It is needed to separate the theories by the introduction of extra variables, called *parameters* and to transfer results from reasoning in $\mathcal{T}_1$ to $\mathcal{T}_2$ and vice versa.

For example, consider a combination of $\mathcal{T}_1 = \{\mathcal{A}_{\mathrm{LRA}}\}$, Section 6.2,

with EUF, $\mathcal{T}_2 = \{\top\}$, Section 6.1 with signatures $\Sigma_1 = \Sigma_{\text{LA}}$ and $\Sigma_2 = (\{S, \text{LA}\}, \{g, a, b, c\}, \emptyset)$ and ground formula

$$\phi = g(b) > 5 \wedge g(c) < 5 \wedge g(c) \approx a \wedge g(b) \approx a.$$

Note that for LRA I fixed the standard algebra, whereas for EUF I fixed a set with one axiom, actually $\top$. So for EUF all first-order $\Sigma_2$-algebras are considered. For both theories Chapter 6 contains decision procedures, however, $\phi$ contains mixed atoms such as $g(b) > 5$ that cannot be processed by the respective decision procedures. So the first step is *purification* where all mixed atoms are translated into pure atoms of $\Sigma_1$, $\Sigma_2$, respectively.

$$\phi = x_{\text{LA}} > 5 \wedge y_{\text{LA}} < 5 \wedge g(c) \approx a \wedge g(b) \approx a \wedge g(b) \approx x_{\text{LA}} \wedge g(c) \approx y_{\text{LA}}$$

Note parameters, e.g., $x_{\text{LA}}$, $y_{\text{LA}}$, are always implicitly existentially quantified. Now the separated formulas considered for both theories are

$$\begin{aligned}\phi_1 &= x_{\text{LA}} > 5 \wedge y_{\text{LA}} < 5\\ \phi_2 &= g(c) \approx a \wedge g(b) \approx a \wedge g(b) \approx x_{\text{LA}} \wedge g(c) \approx y_{\text{LA}}\end{aligned}$$

Any LRA procedure for $\phi_1$ immediately returns true. Congruence closure applied to $\phi_2$ generates $x_{\text{LA}} \approx y_{\text{LA}}$ for the two existentially quantified variables. Transferring this equation to the LRA procedure on $\phi_1 \wedge x_{\text{LA}} = y_{\text{LA}}$ results in false. Therefore, $\phi$ is not satisfiable.

The example exhibits another property required by the respective theories, they have to be *convex*: if a disjunction of equations is the consequence of the theory, actually one equation holds. This property holds for LRA but not for LIA. For example,

$$1 < x_{\text{LIA}} \wedge x_{\text{LIA}} < 4 \models_{\text{LIA}} x_{\text{LIA}} = 2 \vee x_{\text{LIA}} = 3$$

but none of the two single disjuncts is a consequence. Therefore, the Nelson-Oppen combination procedure between LIA and EUF will not be able to detect unsatisfiability of the already purified formulas

$$\begin{aligned}\phi_1 &= 1 < x_{\text{LIA}} \wedge x_{\text{LIA}} < 4 \wedge 1 < y_{\text{LIA}} \wedge y_{\text{LIA}} < 4 \wedge 1 < z_{\text{LIA}} \wedge z_{\text{LIA}} < 4\\ \phi_2 &= x_{\text{LIA}} \not\approx y_{\text{LIA}} \wedge y_{\text{LIA}} \not\approx z_{\text{LIA}} \wedge z_{\text{LIA}} \not\approx x_{\text{LIA}}.\end{aligned}$$

**Definition 7.1.1** (Convex Theory)**.** A theory $\mathcal{T}$ is *convex* if for a conjunction $\phi$ of literals with $\phi \models_{\mathcal{T}} x_1 \approx y_1 \vee \ldots \vee x_n \approx y_n$ then $\phi \models_{\mathcal{T}} x_k \approx y_k$ for some $k$.

Another property needed for the Nelson-Oppen procedure to work is that the theory models always include models with an infinite domain. Consider the two theories

$$\mathcal{T}_1 = \{\forall x, y(x \approx a \vee x \approx b)\}$$

and

$$\mathcal{T}_2 = \{\forall x, y, z.(x \not\approx y \vee x \not\approx z \vee y \not\approx z\}$$

that do not share any signature symbols. Models of $\mathcal{T}_1$ have at most two elements, models of $\mathcal{T}_2$ at least three. So the conjunction $(\mathcal{T}_1 \cup \mathcal{T}_2)$ is already

unsatisfiable. In order to ensure that different models for the respective theory can be combined, the Nelson-Oppen procedure requires the existence of models with infinite cardinality.

**Definition 7.1.2** (Stably-Infinite Theory)**.** A theory $\mathcal{T}$ is *stably-infinite* if for every quantifier-free formula $\phi$, if $\mathcal{T} \models \phi$, then then there exists also a model $\mathcal{A}$ of infinite cardinality, such that $\mathcal{A} \models_{\mathcal{T}} \phi$

**Definition 7.1.3** (Nelson-Oppen Basic Restrictions)**.** Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two theories. Then the *Nelson-Oppen Basic Restrictions* are:

1. There are decision procedures for $\mathcal{T}_1$ and $\mathcal{T}_2$.
2. Each decision procedure returns a complete set of variable identities as consequence of a formula.
3. $\Sigma_1 \cap \Sigma_2 = \emptyset$ except for common sorts.
4. Both theories are convex.
5. $\mathcal{T}_1$ and $\mathcal{T}_2$ are stably-infinite.

Actually, restriction 7.1.3-2 is not needed, because a given finite quantifier-free formula $\phi$ over $\Sigma_1 \cup \Sigma_2$ contains only finitely many different variables. Now instead of putting the burden to identify variables on the decision procedure, all potential variable identifications can be guessed and tested afterwards. The disadvantage of this approach is, of course, that there are exponentially many identifications with respect to a fixed number of variables. Therefore, assuming 7.1.3-2 results in a more efficient procedure and is also supported by many procedures from Section 6. Still I will also formulate the procedure with respect to guessing the identifications, Definition 7.1.6, Proposition 7.1.7, because it enables a more elegant proof of completeness.

Restriction 7.1.3-5 can be further relaxed to assume that the domains of all shared sorts of all models are either infinite or have the same number of elements.

The Nelson-Oppen restrictions and procedure can be extended from two so several theories in the obvious way.

**Example 7.1.4.** $\mathcal{T}_1$ may be LA with the standard LA model over $\mathbb{Q}$ as the only model in $\mathcal{C}_1$ and $\mathcal{T}_2$ is EUF over $\Sigma_2 = \{a, g, f\}$, where $a$ is a constant, $g$ has arity 1 and $f$ arity 2, with all respective term-generated models in $\mathcal{C}_2$.

The goal of the Nelson-Oppen combination procedure is now to decide the satisfiability of a quantifier-free formula $\phi$ over $\Sigma_1 \cup \Sigma_2$. The variables are implicitly existentially quantified. It actually suffices to consider conjunctions of atoms, because for boolean combinations CDCL(NO), Section 7.2, does the job. The first step of the procedure is to apply purification, i.e., transform the formula $\phi$ into a satisfiability equivalent formula $\phi'$ such that no term of an atom in $\phi'$ contains symbols from $\Sigma_1$ and $\Sigma_2$. This can always be achieved by the introduction of fresh variables.

**Example 7.1.5.** Consider the atom $f(x_1, 0) \geq x_3$ with respect to the theories of Example 7.1.4. The satisfiability preserving purified formula for $f(x_1, 0) \geq x_3$ is $x_4 \geq x_3 \wedge x_4 \approx f(x_1, x_5) \wedge x_5 \approx 0$.

Let $N$ be a set of $\Sigma_1 \cup \Sigma_2$ literals interpreted as the conjunction. Then purification amounts to the exhaustive application of the following rule.

**Purify** $N \uplus \{L[t[s]_i]_p\} \Rightarrow_{\text{NO}} N \uplus \{L[t[z]_i]_p, z \approx s\}$
if $t = f(t_1, \ldots, t_n)$, $s = h(s_1, \ldots, s_m)$, the function symbols $f$ and $h$ are from different signatures, $1 \leq i \leq n$, (i.e., $t_i = s$) and $z$ is a fresh variable of appropriate sort

After exhaustive application of Purify to any set $N$ of $\Sigma_1 \cup \Sigma_2$ literals the set $N$ can actually be split into two sets $N = N_1 \cup N_2$ where $N_1$ is build over $\Sigma_1$, $N_2$ is build over $\Sigma_2$ and $N_1$ and $N_2$ only share variables. Variable equations are distributed in both $N_1$ and $N_2$. Now a Nelson-Oppen problem state is a five tuple $(N_1, E_1, N_2, E_2, s)$ with $s \in \{\top, \bot, \text{fail}\}$, the sets $E_1$ and $E_2$ contain variable equations, and $N_1$, $N_2$ literals over the respective signatures, where

| | |
|---|---|
| $(N_1; \emptyset; N_2; \emptyset; \bot)$ | is the start state for some purified set of atoms $N = N_1 \cup N_2$ where the $N_i$ are built from the respective signatures only |
| $(N_1; E_1; N_2; E_2; \text{fail})$ | is a final state, where $N_1 \cup N_2 \cup E_1 \cup E_2$ is unsatisfiable |
| $(N_1; E_1; N_2; E_2; \bot)$ | is an intermediate state, where $N_1 \cup E_2$ and $N_2 \cup E_1$ have to be checked for satisfiability |
| $(N_1; \emptyset; N_2; \emptyset; \top)$ | is a final state, where $N_1 \cup N_2$ is satisfiable |

**Solve** $(N_1; E_1; N_2; E_2; \bot) \Rightarrow_{\text{NO}} (N_1'; E_1'; N_2'; E_2'; \bot)$
if $N_1' = N_1 \cup E_1 \cup E_2$ and $N_2' = N_2 \cup E_1 \cup E_2$ are both $\mathcal{T}_i$-satisfiable, respectively, $E_1'$ are all new variable equations derivable from $N_1'$, $E_2'$ are all new variable equations derivable from $N_2'$ and $E_1' \cup E_2' \neq \emptyset$

**Success** $(N_1; E_1; N_2; E_2; \bot) \Rightarrow_{\text{NO}} (N_1'; \emptyset; N_2'; \emptyset; \top)$
if $N_1' = N_1 \cup E_1 \cup E_2$ and $N_2' = N_2 \cup E_1 \cup E_2$ are both $\mathcal{T}_i$-satisfiable, respectively, $E_1'$ are all new variable equations derivable from $N_1'$, $E_2'$ are all new variable equations derivable from $N_2'$ and $E_1' \cup E_2' = \emptyset$

**Fail** $(N_1; E_1; N_2; E_2; \bot) \Rightarrow_{\text{NO}} (N_1; E_1; N_2; E_2; \text{fail})$
if $N_1' = N_1 \cup E_1 \cup E_2$ or $N_2' = N_2 \cup E_1 \cup E_2$ is $\mathcal{T}_i$-unsatisfiable, respectively

In the definition of the rules all derived equalities between variables are added to $N_1$ and $N_2$ and the decision procedures are always called to test satisfiability and produce new variable equalities. In an implementation this is not needed, a decision procedure needs only to be called if a new equality was derived by the other decision procedure.

The EUF decision procedure can easily be extended to explicitely produce derived variable equalities. For the suggested LA procedures (Fourier-Motzkin, Simplex, Virtual Substitution) this requires some extra work.

As a first example, consider the formula over LA and EUF

$$f(x_1, 0) \geq x_3 \wedge f(x_1, 0) \leq x_3$$

which becomes after purification

$$x_4 \geq x_3 \wedge f(x_1, x_5) \approx x_4 \wedge x_5 \approx 0 \wedge x_6 \leq x_3 \wedge f(x_1, x_5) \approx x_6$$

and the respective NO derivation is

$$\begin{aligned}
&(\{x_4 \geq x_3, x_5 \approx 0, x_6 \leq x_3\}, \emptyset, \{f(x_1, x_5) \approx x_4, f(x_1, x_5) \approx x_6\}, \emptyset, \bot) \\
&\Rightarrow^{\text{Solve}}_{\text{NO}} (\{x_4 \geq x_3, x_5 \approx 0, x_6 \leq x_3\}, \emptyset, \\
&\qquad \{f(x_1, x_5) \approx x_4, f(x_1, x_5) \approx x_6\}, \{x_4 \approx x_6\}, \bot) \\
&\Rightarrow^{\text{Solve}}_{\text{NO}} (\{x_4 \approx x_6, x_4 \geq x_3, x_5 \approx 0, x_6 \leq x_3\}, \{x_4 \approx x_3, x_6 \approx x_3\}, \\
&\qquad \{f(x_1, x_5) \approx x_4, f(x_1, x_5) \approx x_6, x_4 \approx x_6\}, \emptyset, \bot) \\
&\Rightarrow^{\text{Success}}_{\text{NO}} (\{x_4 \approx x_6, x_4 \geq x_3, x_5 \approx 0, x_6 \leq x_3, x_4 \approx x_3, x_6 \approx x_3\}, \emptyset, \\
&\qquad \{f(x_1, x_5) \approx x_4, f(x_1, x_5) \approx x_6, x_4 \approx x_6, x_4 \approx x_3, x_6 \approx x_3\}, \emptyset, \top)
\end{aligned}$$

Note that the Purify rule was applied in the above example in a slightly different way where the variable $x_5$ is shared for both occurrences of the term $f(x_1, 0)$. For an actual implementation, it is desirable to share as many subterms as possible that way.

As a second example, consider the formula over LA and EUF

$$x - y \approx 0 \wedge g(x) \not\approx g(y)$$

which is already purified and the respective NO derivation is

$$\begin{aligned}
&(\{x - y \approx 0\}, \emptyset, \{g(x) \not\approx g(y)\}, \emptyset, \bot) \\
&\Rightarrow^{\text{Solve}}_{\text{NO}} (\{x - y \approx 0\}, \{x \approx y\}, \{g(x) \not\approx g(y)\}, \emptyset, \bot) \\
&\Rightarrow^{\text{Fail}}_{\text{NO}} (\{x - y \approx 0\}, \{x \approx y\}, \{g(x) \not\approx g(y)\}, \emptyset, \text{fail})
\end{aligned}$$

For EUF variable identities are anyway computed by the congruence closure algorithm when computing the equivalence classes by generating a terminating and confluent $R$ (see Section 6.1). However, for LA and, e.g., the simplex algorithm (see Section 6.2.2), it only comes at additional cost to identify variable identities.

**Definition 7.1.6** (Arrangement)**.** Given a (finite) set of parameters $X$, an *arrangement* $A$ over $X$ is a (finite) set of equalities and inequalities over $X$ such that for all $x_1, x_2 \in X$ either $x_1 \approx x_2 \in A$ or $x_1 \not\approx x_2 \in A$.

**Proposition 7.1.7** (Nelson-Oppen modulo Arrangement)**.** Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two theories satisfying the restrictions of Definition 7.1.3 except for restriction 2. Let $\phi$ be a conjunction of literals over $\Sigma_1 \cup \Sigma_2$. Let $N_1$ and $N_2$ be the purified literal sets out of $\phi$. Then $\phi$ is satisfiable iff there is an arrangement $A$ over vars($\phi$) such that $N_1 \cup A$ is $\mathcal{T}_1$-satisfiable and $N_2 \cup A$ is $\mathcal{T}_2$-satisfiable.

Note that it is not sufficient to consider just equalities for some arrangement, because in one theory these equalities might imply further equalities which are then not transferred into the other theory.

**Theorem 7.1.8** (Nelson-Oppen is Sound, Complete and Terminating)**.** Let $\mathcal{T}_1, \mathcal{T}_2$ be two theories satisfying the Nelson-Oppen basic restrictions. Let $\phi$ be a conjunction of literals over $\Sigma_1 \cup \Sigma_2$ and $N_1$, $N_2$ be the result of purifying $\phi$.
(i) All sequences $(N_1; \emptyset; N_2; \emptyset; \bot) \Rightarrow_{\text{NO}}^{*} \ldots$ are finite.
Let $(N_1; \emptyset; N_2; \emptyset; \bot) \Rightarrow_{\text{NO}}^{*} (N_1; E_1; N_2; E_2; s)$ be a derivation with finite state $(N_1; E_1; N_2; E_2; s)$,
(ii) If $s = \text{fail}$ then $\phi$ is unsatisfiable in $\mathcal{T}_1 \cup \mathcal{T}_2$.
(iii) If $s = \top$ then $\phi$ is satisfiable in $\mathcal{T}_1 \cup \mathcal{T}_2$.

*Proof.* (i) The relation $\Rightarrow_{\text{NO}}$ terminates as soon as no new equations are derived or one combination of formulas and equations is unsatisfiable. There are only finitely many different equations over the common variables of $N_1, N_2$, so $\Rightarrow_{\text{NO}}$ terminates.

(ii) Clearly purification preservers satisfiability. The Solve rule only adds logical consequences of the respective theory. Hence, if rule Fail is applicable then clearly $N_1 \cup E_1$ ($N_2 \cup E_2$) is unsatisfiable, hence $\phi$ is not satisfiable. This proves soundness.

(iii) Completeness is more complicated. I show it for the Nelson-Oppen formulation modulo arrangements, Proposition 7.1.7, completeness of $\Rightarrow_{\text{NO}}$ is then implied by convexity of $\mathcal{T}_1, \mathcal{T}_2$. Assume that the theories $\mathcal{T}_1, \mathcal{T}_2$ are given by possibly countably infinite sets of first-order clauses, we also denote by $\mathcal{T}_1, \mathcal{T}_2$. Then $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \{\phi\}$ is unsatisfiable iff $\mathcal{T}_1 \cup \mathcal{T}_2 \cup N_1 \cup N_2$ is unsatisfiable iff $(\mathcal{T}_1 \wedge N_1) \rightarrow (\neg\mathcal{T}_2 \vee \neg N_2)$ is valid. By Craig's interpolation Theorem 3.12.15, there exists a finite set of clauses $H$ such that $\mathcal{T}_1 \wedge N_1 \rightarrow H$ and $H \rightarrow (\neg\mathcal{T}_2 \vee \neg N_2)$, or, reformulated, $(H \wedge \mathcal{T}_2) \rightarrow \neg N_2$. The symbols used in $H$ are common non-variable symbols of $N_1$ and $N_2$. So $H$ is a conjunction of clauses over equations with universally quantified variables $y_j$ and shared parameters $x_i$. It has the form $\bigwedge \bigvee [\neg] t_i \approx t_j$ of equational literals where the $t_i, t_j$ are universally quantified variables $y_j$ or parameters $x_i$. An equation $y_i \approx y_j$ between universally quantified variables is true iff $i = j$ and therefore needs not to be considered. Now this CNF can be transformed into a DNF yielding $\bigvee \bigwedge t_i \approx t_j$, in summary, $(\mathcal{T}_1 \wedge N_1) \rightarrow (\bigvee \bigwedge t_i \approx t_j)$. Next, I prove by contradiction that actually one conjunct $\bigwedge t_i \approx t_j$ is implied and no $t_i, t_j$ is a universally quantified variable. Assume this is not the case, i.e., in each of the conjuncts there are equation(s) $[\neg] x_i \approx a_i$ needed to establish the overall truth of $(\mathcal{T}_1 \wedge N_1) \rightarrow (\bigvee \bigwedge t_i \approx t_j)$. Then $(\mathcal{T}_1 \wedge N_1) \rightarrow (\bigvee x_i \approx a_i)$, where I filter only the positive equations out of

the conjuncts. But the formula $(\bigvee x_i \approx a_i)$ implies a finite model, contradicting that $\mathcal{T}_1$ is stably infinite. Therefore, if $\mathcal{T}_1 \cup \mathcal{T}_2 \cup N_1 \cup N_2$ is unsatisfiable, then there is an arrangement $E$ of the parameters such that $(\mathcal{T}_1 \wedge N_1 \wedge E)$ or $(\mathcal{T}_2 \wedge N_2 \wedge E)$ is unsatisfiable. □

## 7.2 CDCL(T)

Consider a SAT problem where the propositional variables actually stand for ground atoms over some theory $\mathcal{T}$, or a Nelson-Oppen combination of theories, e.g., ground equations or ground atoms of LRA, i.e., LRA atoms where all variables are existentially quantified. The basic idea of all procedures in this section is to apply CDCL, Section 2.9, in order to investigate the boolean structure of the problem. If CDCL derives unsatisfiability, then the problem clearly is. If CDCL derives satisfiability, then a ground decision procedure for $\mathcal{T}$ has to check whether the actual CDCL assignment constitutes also a model in $\mathcal{T}$.

For example, let $\mathcal{T}$ be the purely equational ground theory over free symbols (EUF) where we consider Congruence Closure (Section 6.1) as a decision procedure. Now consider a formula

$$f(a) \approx b \wedge b \approx c \wedge (f(a) \not\approx c \vee a \not\approx c)$$

and its boolean abstraction (clauses)

$$P_1 \wedge P_2 \wedge (P_3 \vee P_4).$$

A CDCL algorithm might find the propositional model $M_1 = P_1 P_2 P_3$. Obviously, the respective literals $f(a) \approx b$, $b \approx c$, $f(a) \not\approx c$ are contradictory in EUF. So $M_1$ does not correspond to a $\mathcal{T}$-model. The congruence closure algorithm can easily justify this contradiction with respect to the literals $P_1, P_2, P_3$, and hence the CDCL algorithm can learn the clause $\neg P_1 \vee \neg P_2 \vee \neg P_3$. Adding this clause to the above clauses

$$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_3)$$

the CDCL algorithm finds the next model $M_2 = P_1 P_2 \neg P_3 P_4$ corresponding to the literals $f(a) \approx b$, $b \approx c$, $f(a) \approx c$, and $a \not\approx c$ which are satisfiable in EUF. So, an overall model is found.

Let $N$ be a finite set of clauses over some theory $\mathcal{T}$ over signature $\Sigma_\mathcal{T}$ such that there exists a decision procedure for satisfiability of a conjunction of literals: $\models_\mathcal{T} L_1 \wedge \ldots \wedge L_n$. Let atr be a bijection from the atoms over $\Sigma_\mathcal{T}$ into propositional variables $\Sigma_{\text{PROP}}$ such that $\text{atr}^{-1}(\text{atr}(A)) = A$. Furthermore, atr distributes over the propositional operators, e.g., $\text{atr}(\neg A) = \neg\, \text{atr}(A)$.

**Lemma 7.2.1** (Correctness of atr)**.** Let $N$ be a set of clauses over some theory $\mathcal{T}$. If $\text{atr}(N) \models \bot$ then $N \models_\mathcal{T} \bot$.