Model-Theoretic Methods for Combining Decision Procedures

Silvio GHILARDI

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano - Italy

VOLOS, JULY 7, 2007

Plan of the Mini-Course

Decision Procedures (for fragment of logical languages, often modulo theories) are at the heart of computer science applications, in various areas ranging from formal verification, knowledge representation, artificial intelligence, and so on.

Plan of the Mini-Course

Decision Procedures (for fragment of logical languages, often modulo theories) are at the heart of computer science applications, in various areas ranging from formal verification, knowledge representation, artificial intelligence, and so on.

Concrete problems often are quite heterogeneous in nature, so that many decision procedures might be needed in the same application.

Plan of the Mini-Course

Decision Procedures (for fragment of logical languages, often modulo theories) are at the heart of computer science applications, in various areas ranging from formal verification, knowledge representation, artificial intelligence, and so on.

Concrete problems often are quite heterogeneous in nature, so that many decision procedures might be needed in the same application.

Modular (i.e *black box*) composition of decision procedures is an highly desirable feature in order to save time and resources. When designing integration/communication interfaces, subtle problems may arise deserving - besides non trivial implementation effort - also careful theoretical foundations.

Plan of the Mini-Course

We plan to make a survey of recent developments in the field, starting from some (hopefully enlightening) motivations arising in the verification area.

- Part 0 :Motivations.
- Part I :Combined Constraint Satisfiability: the disjoint case.
- Part II :Combined Constraint Satisfiability: the non-disjoint case.
- Part III : Combined Word Problems.
- Tomorrow : Combination Techniques in Model-Checking.

Plan of the Mini-Course

We plan to make a survey of recent developments in the field, starting from some (hopefully enlightening) motivations arising in the verification area.

- Part 0 :Motivations.
- Part I :Combined Constraint Satisfiability: the disjoint case.
- Part II :Combined Constraint Satisfiability: the non-disjoint case.
- Part III : Combined Word Problems.
- Tomorrow : Combination Techniques in Model-Checking.

Plan of the Mini-Course

We plan to make a survey of recent developments in the field, starting from some (hopefully enlightening) motivations arising in the verification area.

- Part 0 :Motivations.
- Part I :Combined Constraint Satisfiability: the disjoint case.
- Part II :Combined Constraint Satisfiability: the non-disjoint case.
- Part III : Combined Word Problems.
- Tomorrow : Combination Techniques in Model-Checking.

Plan of the Mini-Course

We plan to make a survey of recent developments in the field, starting from some (hopefully enlightening) motivations arising in the verification area.

- Part 0 :Motivations.
- Part I :Combined Constraint Satisfiability: the disjoint case.
- Part II :Combined Constraint Satisfiability: the non-disjoint case.
- Part III : Combined Word Problems.
- Tomorrow : Combination Techniques in Model-Checking.

Plan of the Mini-Course

We plan to make a survey of recent developments in the field, starting from some (hopefully enlightening) motivations arising in the verification area.

- Part 0 :Motivations.
- Part I :Combined Constraint Satisfiability: the disjoint case.
- Part II :Combined Constraint Satisfiability: the non-disjoint case.
- Part III : Combined Word Problems.
- Tomorrow : Combination Techniques in Model-Checking.

Plan of the Mini-Course

We plan to make a survey of recent developments in the field, starting from some (hopefully enlightening) motivations arising in the verification area.

- Part 0 :Motivations.
- Part I :Combined Constraint Satisfiability: the disjoint case.
- Part II :Combined Constraint Satisfiability: the non-disjoint case.
- Part III : Combined Word Problems.
- Tomorrow : Combination Techniques in Model-Checking.

Part 0

Motivations

§1. A Software Verification Example

An example (taken from a FroCoS 05 paper) shows what is needed in certain applications to the verification area. Consider the following two program fragments written in C language:

```
for (k=1; k<=n; k++)                for (k=1; k<=n; k++)
   a[i+k] = a[i]+k;                    a[i+n-k] = a[i+n]-k;
```

If the execution of either fragment produces the same result in the array a, then `a[i+n]==a[i]+n` must hold initially for any value of `i` and n.

Fixed an integer n, we want to automatically prove the above property.

§1. A Software Verification Example

This amounts to show the unsatisfiability of the conjunction of literals

$$L^n_n = R^n_n \quad \wedge \quad a[i+n] \neq a[i] + n \tag{1}$$

where

$$L^n_k = R^n_k = a \qquad (k = 0)$$

$$L^n_k = wr(L^n_{k-1}, i+k, a[i+k]) \qquad (1 \leq k \leq n)$$

$$R^n_k = wr(R^n_{k-1}, i+n-k, a[i+n-k]) \qquad (1 \leq k \leq n)$$

§1. A Software Verification Example

Satisfiability of (1) has to be checked in the models of the union of Presburger arithmetic and McCarthy's theory of arrays (see p.16 below). Satisfiability of conjunctions of literals is known to be decidable in both theories separately.

§1. A Software Verification Example

Satisfiability of (1) has to be checked in the models of the union of Presburger arithmetic and McCarthy's theory of arrays (see p.16 below). Satisfiability of conjunctions of literals is known to be decidable in both theories separately.

Usually, one needs the check satisfiability not only of conjunctions of literals, but more generally of *quantifier-free formulae.*

§1. A Software Verification Example

Satisfiability of (1) has to be checked in the models of the union of Presburger arithmetic and McCarthy's theory of arrays (see p.16 below). Satisfiability of conjunctions of literals is known to be decidable in both theories separately.

Usually, one needs the check satisfiability not only of conjunctions of literals, but more generally of *quantifier-free formulae.*

Unbounded verification problems are also amenable to a (semi)automatic analysis through satisfiability of quantifier-free formulae by the so called *abstract-check-refine* method, as implemented in tools like the model-checker BLAST.

§2. SMT- tools

Before considering more complex satisfiability problems, let's go back to usual SAT-problems:

§2. SMT- tools

Before considering more complex satisfiability problems, let's go back to usual SAT-problems:

- Input: a classical propositional formula φ.

§2. SMT- tools

Before considering more complex satisfiability problems, let's go back to usual SAT-problems:

- Input: a classical propositional formula φ.
- Output: yes, if there is a boolean assignment satisfying φ; no otherwise.

§2. SMT- tools

Before considering more complex satisfiability problems, let's go back to usual SAT-problems:

- Input: a classical propositional formula φ.
- Output: yes, if there is a boolean assignment satisfying φ; no otherwise.

The problem is known to be NP-complete. By applying linear time structural transformations (no expensive distributive law!), we can assume that φ is a conjunction of clauses.

§2. SMT- tools

Modern SAT-solvers are based on highly optimized variants of classical *DPLL-procedure* (Davis-Putnam 1960 + Davis-Logemann-Loveland 1962).

§2. SMT- tools

Modern SAT-solvers are based on highly optimized variants of classical *DPLL-procedure* (Davis-Putnam 1960 + Davis-Logemann-Loveland 1962).

- perform *deterministic* choices first (unit resolution, backward subsumption, pure literal assignments);

§2. SMT- tools

Modern SAT-solvers are based on highly optimized variants of classical *DPLL-procedure* (Davis-Putnam 1960 + Davis-Logemann-Loveland 1962).

- perform *deterministic* choices first (unit resolution, backward subsumption, pure literal assignments);
- then choose an atom for case-distinction (*semantic* splitting);

§2. SMT- tools

Modern SAT-solvers are based on highly optimized variants of classical *DPLL-procedure* (Davis-Putnam 1960 + Davis-Logemann-Loveland 1962).

- perform *deterministic* choices first (unit resolution, backward subsumption, pure literal assignments);
- then choose an atom for case-distinction (*semantic* splitting);
- make use of appropriate *heuristics*.

§2. SMT- tools

Heuristics include for instance

§2. SMT- tools

Heuristics include for instance

- selection criteria for splitting atoms;

§2. SMT- tools

Heuristics include for instance

- selection criteria for splitting atoms;
- non-chronological backtracking;

§2. SMT- tools

Heuristics include for instance

- selection criteria for splitting atoms;
- non-chronological backtracking;
- conflict-driven learning.

§2. SMT- tools

Heuristics include for instance

- selection criteria for splitting atoms;
- non-chronological backtracking;
- conflict-driven learning.

State-of-the-art SAT-solvers like MINISAT, Z-CHAFF, ... currently handle problems with $\approx 10K$ variables or even more ....

§2. SMT- tools

Heuristics include for instance

- selection criteria for splitting atoms;
- non-chronological backtracking;
- conflict-driven learning.

State-of-the-art SAT-solvers like MINISAT, Z-CHAFF, ... currently handle problems with $\approx 10K$ variables or even more ....

Efficiency of SAT-solvers increased their application domain (e.g. to planning, bounded model-checking, security, etc.). DPLL-based techniques are at the heart of tools for description logics too.

§2. SMT- tools

As we saw, applications often require solving satisfiability problems of quantifier-free formulae *modulo a first-order theory T*.

§2. SMT- tools

As we saw, applications often require solving satisfiability problems of quantifier-free formulae *modulo a first-order theory T*.

Systems implementing such specialized satisfiability problems like Yices, BarcelogicTools, CVC Lite, haRVey, Math-SAT, etc. are called ***S****(atisfiability)* ***M****(odulo)* ***T****(heory)-solvers*.

§2. SMT- tools

As we saw, applications often require solving satisfiability problems of quantifier-free formulae *modulo a first-order theory T*.

Systems implementing such specialized satisfiability problems like Yices, BarcelogicTools, CVC Lite, haRVey, Math-SAT, etc. are called ***S****(atisfiability)* ***M****(odulo)* ***T****(heory)-solvers*.

SMT-competition takes place every year since 2005.

§2. SMT- tools

As we saw, applications often require solving satisfiability problems of quantifier-free formulae *modulo a first-order theory T*.

Systems implementing such specialized satisfiability problems like Yices, BarcelogicTools, CVC Lite, haRVey, Math-SAT, etc. are called ***S****(atisfiability)* ***M****(odulo)* ***T****(heory)-solvers*.

SMT-competition takes place every year since 2005.

In concrete cases T might be the **union** of various theories: the design of appropriate combination algorithms and their properties (soundness, completeness, termination, etc.) is the main concern of the present slides.

§2. SMT- tools

The structure of an SMT-solver is roughly as follows:

§2. SMT- tools

The structure of an SMT-solver is roughly as follows:

- a boolean assignment that is then checked for T-satisfiability is found (lazy approach);

§2. SMT- tools

The structure of an SMT-solver is roughly as follows:

- a boolean assignment that is then checked for T-satisfiability is found (lazy approach);
- the assignment might be partial and checked before splitting (early pruning);

§2. SMT- tools

The structure of an SMT-solver is roughly as follows:

- a boolean assignment that is then checked for T-satisfiability is found (lazy approach);
- the assignment might be partial and checked before splitting (early pruning);
- usual heuristics like non-chronological backtracking and learning are employed.

§2. SMT- tools

The structure of an SMT-solver is roughly as follows:

- a boolean assignment that is then checked for T-satisfiability is found (lazy approach);
- the assignment might be partial and checked before splitting (early pruning);
- usual heuristics like non-chronological backtracking and learning are employed.

DPLL(T) is an example of a formally structured extension of DPLL which is able to cope with the above aspects.

§3. Statement of the Problem

Let T be a first-order theory (in a first-order signature Σ) and let Γ be finite set of Σ-literals.[a] We are asked whether there are a model of T and a variable assignment in it satisfying Γ. We call this the **constraint satisfiability** (CS for short) **problem for** T.

§3. Statement of the Problem

Let T be a first-order theory (in a first-order signature Σ) and let Γ be finite set of Σ-literals.[a] We are asked whether there are a model of T and a variable assignment in it satisfying Γ. We call this the **constraint satisfiability** (CS for short) **problem for** T.

Notice that Γ may contain free variables: it should be clear from above that these variables are meant to be *existentially* (and not universally) quantified. To stress this fact, sometimes variables are replaced by free (i.e. fresh) constants in the constraints to be tested for satisfiability.

---

[a] Equalities among terms and their negations are always included among literals (we consider the identity predicate as a logical constant).

§3. Statement of the Problem

Notice that being able to decide CS problem for T is the same as to be able to decide the universal sentences for T-validity (or again the quantifier-free fragment for T-satisfiability).

§3. Statement of the Problem

Notice that being able to decide CS problem for T is the same as to be able to decide the universal sentences for T-validity (or again the quantifier-free fragment for T-satisfiability).

However, SMT-techniques are needed in concrete implementations to expand decision procedures for CS problem to decision procedures for T-satisfiability of quantifier-free formulae.

§3. Statement of the Problem

Notice that being able to decide CS problem for T is the same as to be able to decide the universal sentences for T-validity (or again the quantifier-free fragment for T-satisfiability).

However, SMT-techniques are needed in concrete implementations to expand decision procedures for CS problem to decision procedures for T-satisfiability of quantifier-free formulae.

In particular, modules for T-constraint satisfiability are used by SMT-solvers when checking (partial) assignments found by the propositional SAT enumerator.

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

- the empty theory (here CS is the so-called congruence closure problem);

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

- the empty theory (here CS is the so-called congruence closure problem);
- linear (rational or integer) arithmetic;

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

- the empty theory (here CS is the so-called congruence closure problem);
- linear (rational or integer) arithmetic;
- theories axiomatizing common datatypes (lists, arrays, ...);

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

- the empty theory (here CS is the so-called congruence closure problem);
- linear (rational or integer) arithmetic;
- theories axiomatizing common datatypes (lists, arrays, ...);
- theories coming from computer algebra (K-algebras, ...);

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

- the empty theory (here CS is the so-called congruence closure problem);
- linear (rational or integer) arithmetic;
- theories axiomatizing common datatypes (lists, arrays, ...);
- theories coming from computer algebra (K-algebras, ...);
- algebraic counterparts of modal logics (i.e. theories axiomatizing varieties of Boolean algebras with operators).

§4. Useful Theories

There are many examples of theories in which CS problem is solvable:

- the empty theory (here CS is the so-called congruence closure problem);
- linear (rational or integer) arithmetic;
- theories axiomatizing common datatypes (lists, arrays, ...);
- theories coming from computer algebra (K-algebras, ...);
- algebraic counterparts of modal logics (i.e. theories axiomatizing varieties of Boolean algebras with operators).

Notice that in all the above cases there is a big gap in decidability/complexity between satisfiability of quantifier-free and arbitrary first order formulae.

§4. Useful Theories

McCarthy's theory of *arrays* has three sorts (for arrays, index and elements, respectively); axioms are the following:

- $wr(a, i, e)[i] = e$;
- $wr(a, i, e)[j] = a[j] \vee i = j$;
- $a = b \leftrightarrow \forall i\ (a[i] = b[i])$.

(the last is called the extensionality axiom). Whereas the full first-order decision problem for this theory is undecidable, the quantifier-free fragment satisfiability is just NP.

§4. Useful Theories

The theory of *acyclic lists* is axiomatized as follows:

- $car(cons(x, y)) = x$;
- $cdr(cons(x, y)) = y$;
- $cons(car(x), cdr(x)) = x$;
- $x \neq t(x)$, where t consists of a (non empty) series of applications of car, cdr in any order.

Again, constraint satisfiability is decidable in linear time, whereas full first-order satisfiability is not elementary.

Suggested Readings:

- **On Abstract-Check-Refine**:

  [1] T. Henzinger, R. Jhala, R. Majumdar, K. McMillan *Abstractions from Proofs*, Proceedings of POPL ’04, ACM Press.

- **On Satisfiability Modulo Theories**:

  [2] R. Nieuwhenhuis, A. Oliveras, C. Tinelli *Solving SAT and SAT modulo theories: from an abstract DPLL to DPLL(T)*, Journal of the ACM (to appear) [available from authors’ web pages]

  [3] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Ranise, R. Sebastiani. *Efficient theory combination via boolean search*, Information and Computation, 204(10), pp. 1493-1525, 2006.

Suggested Readings:

- **On Congruence Closure**:

  [4] R. Nieuwenhuis, A. Oliveras *Fast Congruence Closure and Extensions*, Information and Computation, 205(4):557-580, 2007.

- **On Numerical Constraints**:

  [5] Bockmayr A., Weispfenning V., *Solving Numerical Constraints*, in Robinson A., Voronkov A., (eds.) “Handbook of Automated Reasoning”, vol. I, Elsevier/MIT, pp. 751-842 (2001).

Suggested Readings:

- **On Arrays**:

[6] A. R. Bradley, Z. Manna, and H. B. Sipma. *What's decidable about arrays?*, Proc. of VMCAI'06, vol. 3855 of *LNCS*, 2006.

[7] S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. *Deciding extension of the theory of arrays by integrating decision procedures and instantiation strategies.* Proc. of JELIA 06, vol. 4160 of *LNAI*, 2006.

Suggested Readings:

- **On Software Verification Theories** (with focus on combination):

[8] D. Oppen *Complexity, Convexity, and Combination of Theories*, Theor. Comp. Sc, 12, pp. 291-302, 1980.

[9] S. Ranise, C. Ringeissen, D.K. Tran, *Nelson-Oppen, Shostak and the Extended Canonizer: A Family Picture with a Newborn*, Proc. ICTAC ’04, *LNCS*, 2004.

[10] S. Krstic, A. Goel, J, Grundy, C, Tinelli *Combined Satisfiability Modulo Parametric Theories*, Proc. TACAS’07, *LNCS*, 2007.

Part I

Combined CS: the Disjoint Case

§1. Combined CS: the Disjoint Case

Our main task: given algorithms for deciding constraint satisfiability in two theories T_1, T_2 (over signatures[a] Σ_1, Σ_2), **how to build an algorithm for deciding constraint satisfiability in** $T_1 \cup T_2$?

§1. Combined CS: the Disjoint Case

Our main task: given algorithms for deciding constraint satisfiability in two theories T_1, T_2 (over signatures[a] Σ_1, Σ_2), **how to build an algorithm for deciding constraint satisfiability in** $T_1 \cup T_2$?

We shall illustrate some techniques for this combination problem. Once again, such techniques need suitable 're-engineering' in the SMT-solvers case, where Boolean combinations of atoms (and not just constraints) must be taken into account.

[a] All our signatures are at most countable.

§1. Combined CS: the Disjoint Case

Let us indicate by Σ_0 the common subsignature $\Sigma_1 \cap \Sigma_2$ and let us first analyze the following simpler case:

§1. Combined CS: the Disjoint Case

Let us indicate by Σ_0 the common subsignature $\Sigma_1 \cap \Sigma_2$ and let us first analyze the following simpler case:

- Σ_0 **is empty** (i.e. variable equations and inequations are the only possible Σ_0-literals).

§1. Combined CS: the Disjoint Case

Let us indicate by Σ_0 the common subsignature $\Sigma_1 \cap \Sigma_2$ and let us first analyze the following simpler case:

- Σ_0 **is empty** (i.e. variable equations and inequations are the only possible Σ_0-literals).

This is the case originally considered by Nelson-Oppen in 1979.

§1. Combined CS: the Disjoint Case

Warning. There *cannot be* a general effective method for combining decision procedures leading always to a complete algorithm:

§1. Combined CS: the Disjoint Case

Warning. There *cannot be* a general effective method for combining decision procedures leading always to a complete algorithm:

$\boxed{\text{Theorem 0.}}$ [Bonacina, Ghilardi, Ranise, Nicolini and Zucchelli, IJCAR 06] *There are theories* T_1, T_2 *having disjoint signatures and decidable CS problem such that CS problem in* $T_1 \cup T_2$ *is undecidable.*

§1. Combined CS: the Disjoint Case

Warning. There *cannot be* a general effective method for combining decision procedures leading always to a complete algorithm:

$\boxed{\text{Theorem 0.}}$ [Bonacina, Ghilardi, Ranise, Nicolini and Zucchelli, IJCAR 06] *There are theories T_1, T_2 having disjoint signatures and decidable CS problem such that CS problem in $T_1 \cup T_2$ is undecidable.*

Reason for this negative result: the fact that you are able to decide whether a Σ_1-constraint Γ_1 is satisfiable in a model of T_1 does not mean that you are able to decide whether it is satisfiable in an *infinite* model of T_1. However, if T_2 has only infinite models, deciding satisfiability of Γ_1 modulo $T_1 \cup T_2$ requires precisely that.

§2. The Nelson-Oppen Method

Nelson-Oppen method (Nelson-Oppen, 1979) is the most simple method for combining decision procedures for constraint satisfiability. It was originally proposed for disjoint (first-order) signatures, but it can be applied in a broader context. We summarize here the essence of Nelson-Oppen method from an *intuitive* point of view.

§2. The Nelson-Oppen Method

Nelson-Oppen method (Nelson-Oppen, 1979) is the most simple method for combining decision procedures for constraint satisfiability. It was originally proposed for disjoint (first-order) signatures, but it can be applied in a broader context. We summarize here the essence of Nelson-Oppen method from an *intuitive* point of view.

Let $T_1, T_2, \Sigma_1, \Sigma_2, \Sigma_0$ be as above (Σ_0 is the common subsignature which is empty and constraint satisfiability is decidable in T_1, T_2); we fix also a finite set of $\Sigma_1 \cup \Sigma_2$-literals Γ.

§2. The Nelson-Oppen Method

Nelson-Oppen method (Nelson-Oppen, 1979) is the most simple method for combining decision procedures for constraint satisfiability. It was originally proposed for disjoint (first-order) signatures, but it can be applied in a broader context. We summarize here the essence of Nelson-Oppen method from an *intuitive* point of view.

Let $T_1, T_2, \Sigma_1, \Sigma_2, \Sigma_0$ be as above (Σ_0 is the common subsignature which is empty and constraint satisfiability is decidable in T_1, T_2); we fix also a finite set of $\Sigma_1 \cup \Sigma_2$-literals Γ.

Checking satisfiability of $T_1 \cup T_2 \cup \Gamma$ by Nelson-Oppen requires the following phases:

§2. The Nelson-Oppen Method

- $\boxed{\text{Purification}}$: an equi-satisfiable pure constraint $\Gamma_1 \cup \Gamma_2$ is produced (this is achieved by Purification Rule, replacing a subterm t by a fresh variables x - the equation $x = t$ is also added to the current constraint); we let $\underline{x}_0$ be the variables occurring in $\Gamma_1 \cup \Gamma_2$.

§2. The Nelson-Oppen Method

- $\boxed{\text{Purification}}$: an equi-satisfiable pure constraint $\Gamma_1 \cup \Gamma_2$ is produced (this is achieved by Purification Rule, replacing a subterm t by a fresh variables x - the equation $x = t$ is also added to the current constraint); we let $\underline{x}_0$ be the variables occurring in $\Gamma_1 \cup \Gamma_2$.
- $\boxed{\text{Propagation}}$: the T_1-constraint satisfiability procedure and the T_2-constraint satisfiability procedure fairly exchange information concerning entailed unsatisfiability of Σ_0-constraints in which at most the variables $\underline{x}_0$ occur.

§2. The Nelson-Oppen Method

- $\boxed{\text{Purification}}$: an equi-satisfiable pure constraint $\Gamma_1 \cup \Gamma_2$ is produced (this is achieved by Purification Rule, replacing a subterm t by a fresh variables x - the equation $x = t$ is also added to the current constraint); we let $\underline{x}_0$ be the variables occurring in $\Gamma_1 \cup \Gamma_2$.
- $\boxed{\text{Propagation}}$: the T_1-constraint satisfiability procedure and the T_2-constraint satisfiability procedure fairly exchange information concerning entailed unsatisfiability of Σ_0-constraints in which at most the variables $\underline{x}_0$ occur.
- $\boxed{\text{Until}}$: an inconsistency is detected or a saturation state is reached.

§2. The Nelson-Oppen Method

To make the above schema more precise, we need some observations:

§2. The Nelson-Oppen Method

To make the above schema more precise, we need some observations:

- $\boxed{\text{About Purification}}$: this is not problematic and requires only linear time;

§2. The Nelson-Oppen Method

To make the above schema more precise, we need some observations:

- $\boxed{\text{About Purification}}$: this is not problematic and requires only linear time;
- $\boxed{\text{About Propagation}}$: this is also not problematic, but see below;

§2. The Nelson-Oppen Method

To make the above schema more precise, we need some observations:

- $\boxed{\text{About Purification}}$: this is not problematic and requires only linear time;
- $\boxed{\text{About Propagation}}$: this is also not problematic, but see below;
- $\boxed{\text{About the Exit from the Loop}}$: whereas it is evident that the procedure is sound (if an inconsistency is detected the input constraint is unsatisfiable), there is no guarantee at all about *completeness*, in other words reaching saturation does not imply consistency. By the above undecidability result, we know that we need conditions to ensure completeness.

§3. Propagation

We can implement Propagation in two ways (notice that Σ_0-atoms are variable equations):

- $\boxed{\text{Propagation (Guessing Version)}}$: here we simply make a *guess* of a $\Sigma_0(\underline{x}_0)$-arrangement (namely we guess for a maximal set of Σ_0-literals containing at most the variables $\underline{x}_0$) and check it for both $T_1 \cup \Gamma_1$-consistency and $T_2 \cup \Gamma_2$-consistency.

§3. Propagation

We can implement Propagation in two ways (notice that Σ_0-atoms are variable equations):

- $\boxed{\text{Propagation (Guessing Version)}}$: here we simply make a *guess* of a $\Sigma_0(\underline{x}_0)$-arrangement (namely we guess for a maximal set of Σ_0-literals containing at most the variables $\underline{x}_0$) and check it for both $T_1 \cup \Gamma_1$-consistency and $T_2 \cup \Gamma_2$-consistency.

- $\boxed{\text{Propagation (Backtracking Version)}}$: identify a disjunction of $\underline{x}_0$*-atoms* $A_1 \vee \cdots \vee A_n$ which is entailed by $T_i \cup \Gamma_i$ ($i = 1$ or 2) and make case splitting by adding some A_j to both Γ_1, Γ_2 (if none of the $A_1, \dots, A_n$ is already there). Repeat until possible.

§3. Propagation

- An advantage of the first option is that whenever constraints are represented not as sets of literals, but as boolean combinations of atoms, one may combine heuristics of SMT-solvers with specific features of the theories to be combined in order to produce efficiently the right arrangement.

§3. Propagation

- An advantage of the first option is that whenever constraints are represented not as sets of literals, but as boolean combinations of atoms, one may combine heuristics of SMT-solvers with specific features of the theories to be combined in order to produce efficiently the right arrangement.
- An advantage of the second option is that it works in the non disjoint case under *noetherianity* hypotheses (we turn to this later on).

§3. Propagation

- Another advantage of the second method is that the procedure can be made deterministic in case the T_i are both Σ_0-*convex* (T_i is said to be Σ_0-convex iff whenever $T_i \cup \Gamma_i$ entails a disjunction of $n > 1$ Σ_0-atoms, then it entails one of them).

§3. Propagation

- Another advantage of the second method is that the procedure can be made deterministic in case the T_i are both Σ_0-*convex* (T_i is said to be Σ_0-convex iff whenever $T_i \cup \Gamma_i$ entails a disjunction of $n > 1$ Σ_0-atoms, then it entails one of them).

Universal Horn theories are Σ_0-convex; by using simple properties of convex sets, we can show that real linear arithemtic is Σ_0-convex (this case explains the reason for the name ‘convex’).

§3. Propagation

- Another advantage of the second method is that the procedure can be made deterministic in case the T_i are both Σ_0-*convex* (T_i is said to be Σ_0-convex iff whenever $T_i \cup \Gamma_i$ entails a disjunction of $n > 1$ Σ_0-atoms, then it entails one of them).

Universal Horn theories are Σ_0-convex; by using simple properties of convex sets, we can show that real linear arithemtic is Σ_0-convex (this case explains the reason for the name ‘convex’).

From the complexity viewpoint, convexity may keep combined problems tractable, since it avoids don’t-know nondeterminism.

§4. Completeness

The standard requirement to gain completeness is stably infiniteness: a theory T is said to be *stably infinite* iff every T-satisfiable constraint is satisfiable in an infinite model of T (by compactness, this is the same as requiring that every model of T embeds into an infinite model of T).

§4. Completeness

The standard requirement to gain completeness is stably infiniteness: a theory T is said to be *stably infinite* iff every T-satisfiable constraint is satisfiable in an infinite model of T (by compactness, this is the same as requiring that every model of T embeds into an infinite model of T).

$\boxed{\text{Theorem 1.}}$ *If T_1, T_2 are both stably infinite and the shared subsignature Σ_0 is empty, then Nelson-Oppen procedure transfers decidability of constraint satisfiability problems from T_1 and T_2 to $T_1 \cup T_2$.*

§5. Asymmetric Approaches

Stable infiniteness requirement is sometimes a real drawback (e.g. enumerated datatypes theories are not stably infinite!) To overcome it, asymmetric approaches have been proposed: in these approaches, different kind of requirements are asked for T_1 and T_2.

§5. Asymmetric Approaches

Stable infiniteness requirement is sometimes a real drawback (e.g. enumerated datatypes theories are not stably infinite!) To overcome it, asymmetric approaches have been proposed: in these approaches, different kind of requirements are asked for T_1 and T_2.

The Nelson-Oppen combination schema is slightly modified accordingly.

§5. Asymmetric Approaches

Stable infiniteness requirement is sometimes a real drawback (e.g. enumerated datatypes theories are not stably infinite!) To overcome it, asymmetric approaches have been proposed: in these approaches, different kind of requirements are asked for T_1 and T_2.

The Nelson-Oppen combination schema is slightly modified accordingly.

We give here few more information on these asymmetric approaches, which are rather simple but sometimes amazingly powerful.

§5. Asymmetric Approaches

A theory T in the signature Σ is said to be *shiny* iff for every T-satisfiable constraint Γ it is possible to compute a finite cardinal κ such that Γ has a T-model in every cardinality $\lambda \geq \kappa$.

§5. Asymmetric Approaches

A theory T in the signature Σ is said to be *shiny* iff for every T-satisfiable constraint Γ it is possible to compute a finite cardinal κ such that Γ has a T-model in every cardinality $\lambda \geq \kappa$.

$\boxed{\text{Theorem 2.}}$ [Tinelli-Zarba, 03] *If T_1 is shiny and the shared subsignature Σ_0 is empty, then decidability of constraint satisfiability problems transfers from T_1 and T_2 to $T_1 \cup T_2$.*

§5. Asymmetric Approaches

A theory T in the signature Σ is said to be *shiny* iff for every T-satisfiable constraint Γ it is possible to compute a finite cardinal κ such that Γ has a T-model in every cardinality $\lambda \geq \kappa$.

$\boxed{\text{Theorem 2.}}$ [Tinelli-Zarba, 03] *If T_1 is shiny and the shared subsignature Σ_0 is empty, then decidability of constraint satisfiability problems transfers from T_1 and T_2 to $T_1 \cup T_2$.*

Since the pure equality theory in any signature is shiny, we get:

§5. Asymmetric Approaches

A theory T in the signature Σ is said to be *shiny* iff for every T-satisfiable constraint Γ it is possible to compute a finite cardinal κ such that Γ has a T-model in every cardinality $\lambda \geq \kappa$.

Theorem 2. [Tinelli-Zarba, 03] *If T_1 is shiny and the shared subsignature Σ_0 is empty, then decidability of constraint satisfiability problems transfers from T_1 and T_2 to $T_1 \cup T_2$.*

Since the pure equality theory in any signature is shiny, we get:

Corollary 3. [Ganzinger, 2002] *If T is any Σ-theory, then decidability of constraint satisfiability problems transfers from T to any free extension of T in a larger signature $\Omega \supseteq \Sigma$.*

§5. Asymmetric Approaches

In verification one often needs combinations of a theory modeling the *elements* with (one or more) many-sorted theories (such as lists, arrays, sets, multisets, etc.) describing container based *data-structures*. Whereas the theory describing the elements is rather arbitrary, the theory modeling data-structures can be subject to restrictions, provided these restrictions are met in concretely used cases.

§5. Asymmetric Approaches

In verification one often needs combinations of a theory modeling the *elements* with (one or more) many-sorted theories (such as lists, arrays, sets, multisets, etc.) describing container based *data-structures*. Whereas the theory describing the elements is rather arbitrary, the theory modeling data-structures can be subject to restrictions, provided these restrictions are met in concretely used cases.

This is the genuine motivation for taking the asymmetric approach. The same motivation leads to extensions to the many-sorted case (see the notion of *politeness* in [Ranise, Ringeissen, Zarba 05]) and also to higher-order contexts (see the notion of *parametricity* in [Krstic, Goel, Grundy, Tinelli 07]).

§5. Asymmetric Approaches

To complete the picture, we must mention that another combination schema (requiring the existence of so-called ‘solvers’ and ‘canonizers’ for the ingredient theories) has been proposed by Shostak in 1984.

§5. Asymmetric Approaches

To complete the picture, we must mention that another combination schema (requiring the existence of so-called ‘solvers’ and ‘canonizers’ for the ingredient theories) has been proposed by Shostak in 1984.

Despite the original paper suffered by various technical drawbacks, Shostak’s point of view has been quite influential in the implementations.

§5. Asymmetric Approaches

To complete the picture, we must mention that another combination schema (requiring the existence of so-called ‘solvers’ and ‘canonizers’ for the ingredient theories) has been proposed by Shostak in 1984.

Despite the original paper suffered by various technical drawbacks, Shostak’s point of view has been quite influential in the implementations.

Nowadays correct approaches like [Ganzinger 2002] clarified the matter and tend to see Shostak procedure as a refinement of the Nelson-Oppen one.

References:

[1] G. Nelson, D. C. Oppen *Simplification by cooperating decision procedures*, ACM Trans. on Programming Languages and Systems, 1979.

[2] M. P. Bonacina, S. Ghilardi, E. Nicolini, S. Ranise, D. Zucchelli *Decidability and undecidability results for Nelson-Oppen and rewrite-based decision procedures*, Proc. of IJCAR 2006, vol. 4130 of *LNAI*, 2006.

[3] H. Ganzinger, *Shostak Light*, Proc. of CADE 18, Vol. 2392 of *LNAI*, 2002.

[4] C. Tinelli, C. Zarba. *Combining non-stably infinite theories*, Proc. of FTP'03, 2003 (to app. also in the Journal of Automated Reasoning).

[5] S. Ranise, C. Ringeissen, C. Zarba *Combining Data Structures with Nonstably Infinite Theories Using Many-Sorted Logic*, Proc. of FroCoS '05, vol. 3717 of *LNAI*, 2005.

[6] S. Krstic, A. Goel, J. Grundy, C. Tinelli *Combined Satisfiability Modulo Parametric Theories*, Proc. TACAS'07, *LNCS*, 2007.

Part II

Combined CS: the Non-Disjoint Case

§1. Nelson-Oppen Schema Revisited

Let T_1 be a Σ_1-theory and T_2 be a Σ_2-theory; now the common subsignature $\Sigma_0 := \Sigma_1 \cap \Sigma_2$ is not assumed to be empty anymore.

We nevertheless try to apply the (plain symmetric) Nelson-Oppen combination schema:

§1. Nelson-Oppen Schema Revisited

Let T_1 be a Σ_1-theory and T_2 be a Σ_2-theory; now the common subsignature $\Sigma_0 := \Sigma_1 \cap \Sigma_2$ is not assumed to be empty anymore.

We nevertheless try to apply the (plain symmetric) Nelson-Oppen combination schema:

- $\boxed{\text{Purification}}$: no problem, goes as in the disjoint case (but further optimizations are possible);

§1. Nelson-Oppen Schema Revisited

Let T_1 be a Σ_1-theory and T_2 be a Σ_2-theory; now the common subsignature $\Sigma_0 := \Sigma_1 \cap \Sigma_2$ is not assumed to be empty anymore.

We nevertheless try to apply the (plain symmetric) Nelson-Oppen combination schema:

- $\boxed{\text{Purification}}$: no problem, goes as in the disjoint case (but further optimizations are possible);
- $\boxed{\text{Propagation}}$: how to do it in a terminating way ??

§1. Nelson-Oppen Schema Revisited

Let T_1 be a Σ_1-theory and T_2 be a Σ_2-theory; now the common subsignature $\Sigma_0 := \Sigma_1 \cap \Sigma_2$ is not assumed to be empty anymore.

We nevertheless try to apply the (plain symmetric) Nelson-Oppen combination schema:

- $\boxed{\text{Purification}}$: no problem, goes as in the disjoint case (but further optimizations are possible);
- $\boxed{\text{Propagation}}$: how to do it in a terminating way ??
- $\boxed{\text{Completeness}}$: even more problematic than before ...

§1. Nelson-Oppen Schema Revisited

The most simple method for avoiding the non-termination risk is to assume that there is a Σ_0-theory T_0 contained in both T_1, T_2 which is *effectively locally finite*: this means that Σ_0 is finite and that, given a finite set of variables $\underline{x}_0$, there are only finitely many $\Sigma_0(\underline{x}_0)$-terms up to T_0-equivalence. Representative terms for each equivalence class should also be computable.

§1. Nelson-Oppen Schema Revisited

The most simple method for avoiding the non-termination risk is to assume that there is a Σ_0-theory T_0 contained in both T_1, T_2 which is *effectively locally finite*: this means that Σ_0 is finite and that, given a finite set of variables $\underline{x}_0$, there are only finitely many $\Sigma_0(\underline{x}_0)$-terms up to T_0-equivalence. Representative terms for each equivalence class should also be computable.

If effective local finiteness of a shared theory T_0 is assumed, the total amount of exchangeable information is finite. Propagation can be still implemented by guessing (guess a maximal set of representative $\Sigma_0(\underline{x}_0)$-literals) or by backtracking (make case-split on disjunctions of $\Sigma_0(\underline{x}_0)$-atoms that are not entailed by both current purified constraints).

§1. Nelson-Oppen Schema Revisited

We still have to identify sufficient conditions for completeness. To this aim it is sufficient to analyze carefully *from a model-theoretic point of view* the stable infiniteness requirement and the completeness proof in the disjoint case.

§1. Nelson-Oppen Schema Revisited

We still have to identify sufficient conditions for completeness. To this aim it is sufficient to analyze carefully *from a model-theoretic point of view* the stable infiniteness requirement and the completeness proof in the disjoint case.

T_i to be stably infinite means that every model of T_i embeds into a model of $T_i \cup T_0^*$, where T_0^* is the model completion the pure theory of equality T_0.

§1. Nelson-Oppen Schema Revisited

We still have to identify sufficient conditions for completeness. To this aim it is sufficient to analyze carefully *from a model-theoretic point of view* the stable infiniteness requirement and the completeness proof in the disjoint case.

T_i to be stably infinite means that every model of T_i embeds into a model of $T_i \cup T_0^*$, where T_0^* is the model completion the pure theory of equality T_0.

If we adapt this hypothesis to the non-disjoint case (see below for a precise formulation), the completeness proof still works: Robinson Joint Consistency Lemma becomes the main ingredient of it.

§1. Nelson-Oppen Schema Revisited

$\boxed{\text{Definition.}}$ *Let* $T_0 \subseteq T$ *be theories in signatures* $\Sigma_0 \subseteq \Sigma$*; suppose also that* T_0 *is universal and has a model completion* T_0^**. We say that* T *is* T_0*-compatible iff every model of* T *embeds into a model of* $T \cup T_0^*$*.*

§1. Nelson-Oppen Schema Revisited

$\boxed{\text{Definition.}}$ *Let $T_0 \subseteq T$ be theories in signatures $\Sigma_0 \subseteq \Sigma$; suppose also that T_0 is universal and has a model completion T_0^*. We say that T is T_0-compatible iff every model of T embeds into a model of $T \cup T_0^*$.*

We recall that T_0^* being a model completion of $T_0 \subseteq T_0^*$ means that: (i) every model of T_0 embeds into a model of T_0^*; (ii) the union of T_0^* with the Robinson diagram of a model of T_0 is a complete theory.

§1. Nelson-Oppen Schema Revisited

$\boxed{\text{Definition.}}$ *Let $T_0 \subseteq T$ be theories in signatures $\Sigma_0 \subseteq \Sigma$; suppose also that T_0 is universal and has a model completion T_0^*. We say that T is T_0-compatible iff every model of T embeds into a model of $T \cup T_0^*$.*

We recall that T_0^* being a model completion of $T_0 \subseteq T_0^*$ means that: (i) every model of T_0 embeds into a model of T_0^*; (ii) the union of T_0^* with the Robinson diagram of a model of T_0 is a complete theory.

Examples can be easily found in standard model theory textbooks.

§1. Nelson-Oppen Schema Revisited

We are now ready for a first formulation of the combination theorem in the non-disjoint case:

§1. Nelson-Oppen Schema Revisited

We are now ready for a first formulation of the combination theorem in the non-disjoint case:

$\boxed{\text{Theorem 4.}}$ [Ghilardi 04] *Suppose that there is an effectively locally finite and universal Σ_0-subtheory T_0 of T_1 and T_2 which also admits a model completion. If T_1, T_2 are both T_0-compatible, then Nelson-Oppen procedure transfers decidability of constraint satisfiability problem from T_1 and T_2 to $T_1 \cup T_2$.*

§1. Nelson-Oppen Schema Revisited

As a Corollary of the above Theorem, one can easily deduce the *decidability transfer result for global consequence relation to fusions of modal logics* [Wolter 1998].

§1. Nelson-Oppen Schema Revisited

As a Corollary of the above Theorem, one can easily deduce the *decidability transfer result for global consequence relation to fusions of modal logics* [Wolter 1998].

This is because: (i) deciding global consequence relation in a modal logic means deciding constraint satisfiability in the corresponding variety of Boolean algebras with operators; (ii) fusion of modal logics corresponds to union of the equational theories axiomatizing such varieties; (iii) any equational theory axiomatizing a variety of Boolean algebras with operators is BA-compatible (here BA is the theory of Boolean algebras).

§2. Termination by Noetherianity.

The local finiteness requirement ensures termination of the Nelson-Oppen algorithm. If we implement Propagation by backtracking, we can get termination by a requirement that is weaker than local finiteness:

§2. Termination by Noetherianity.

The local finiteness requirement ensures termination of the Nelson-Oppen algorithm. If we implement Propagation by backtracking, we can get termination by a requirement that is weaker than local finiteness:

Definition. *A* Σ_0*-theory* T_0 *is* Noetherian *if and only if for every* finite *set of variables* $\underline{x}_0$*, every infinite ascending chain*

$$\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_n \subseteq \cdots$$

of sets of $\Sigma_0(\underline{x}_0)$*-atoms is eventually constant modulo* T_0 *(i.e. there is an* n *such that* $T_0 \models \bigwedge \Theta_n \to A$*, for every natural number* m *and atom* $A \in \Theta_m$*).*

§2. Termination by Noetherianity.

The above definition is suggested by algebraic examples.

Typically, if T_0 is any equational theory axiomatizing a variety in which finitely generated algebras are finitely presented, then T is noetherian.

Thus, the theory of K-algebras (for a field K), of R-modules (for a noetherian ring R), of abelian groups and semigroups, etc. are noetherian (for applications to verification, this means in particular that linear - integer or real - arithmetic is noetherian, provided ordering is dropped in the signature).

§2. Termination by Noetherianity.

The above definition is suggested by algebraic examples.

Typically, if T_0 is any equational theory axiomatizing a variety in which finitely generated algebras are finitely presented, then T is noetherian.

Thus, the theory of K-algebras (for a field K), of R-modules (for a noetherian ring R), of abelian groups and semigroups, etc. are noetherian (for applications to verification, this means in particular that linear - integer or real - arithmetic is noetherian, provided ordering is dropped in the signature).

An argument based on König Lemma shows that Propagation (implemented through backtracking) must eventually halt if T_0 is noetherian.

§2. Termination by Noetherianity.

However, *it is not true* that Noetherianity of T_0 and T_0-compatibility of both T_1, T_2 are sufficient for a decidability transfer result (there are counterexamples: the trouble is that one may not be able to realize that Propagation is over).

§2. Termination by Noetherianity.

However, *it is not true* that Noetherianity of T_0 and T_0-compatibility of both T_1, T_2 are sufficient for a decidability transfer result (there are counterexamples: the trouble is that one may not be able to realize that Propagation is over).

$\boxed{\text{Definition.}}$ *Let $T_0 \subseteq T$ be theories in signatures $\Sigma_0 \subseteq \Sigma$; given a Σ-constraint Γ and a finite set of free variables $\underline{x}_0$, a T_0-*basis *for Γ w.r.t. $\underline{x}_0$ is a finite set Δ of* positive *$\Sigma_0(\underline{x}_0)$-clauses such that*

- $T \models \bigwedge \Gamma \to C$, *for all $C \in \Delta$ and*
- *if $T \models \bigwedge \Gamma \to C$ then $T_0 \models \bigwedge \Delta \to C$, for every positive $\Sigma_0(\underline{x}_0)$-clause C.*

§2. Termination by Noetherianity.

If T_0 is noetherian, one can prove that T_0-bases exist for every Γ, but we must ensure that they are computable:

§2. Termination by Noetherianity.

If T_0 is noetherian, one can prove that T_0-bases exist for every Γ, but we must ensure that they are computable:

$\boxed{\text{Definition.}}$ *Let $T_0 \subseteq T$ be theories in signatures $\Sigma_0 \subseteq \Sigma$; T is an* effectively Noetherian extension of T_0 *if and only if T_0 is Noetherian and T_0-bases are computable (for all Γ and $\underline{x}_0$).*

§2. Termination by Noetherianity.

If T_0 is noetherian, one can prove that T_0-bases exist for every Γ, but we must ensure that they are computable:

$\boxed{\text{Definition.}}$ *Let $T_0 \subseteq T$ be theories in signatures $\Sigma_0 \subseteq \Sigma$; T is an* effectively Noetherian extension of T_0 *if and only if T_0 is Noetherian and T_0-bases are computable (for all Γ and $\underline{x}_0$).*

When 'good' decision procedures (e.g. decision procedures based on some rewriting/completion mechanism) are available for constraint satisfiability in T, then one may extract T_0-bases out of them (such an extraction might however require little extra work).

§2. Termination by Noetherianity.

We supply here some examples (see [Nicolini, 2006] for details):

§2. Termination by Noetherianity.

We supply here some examples (see [Nicolini, 2006] for details):

- standard Gröbner basis computation (if the admissible order satisfies mild assumptions) shows that the theory of K-algebras is effectively noetherian over the theory of K-vector spaces;

§2. Termination by Noetherianity.

We supply here some examples (see [Nicolini, 2006] for details):

- standard Gröbner basis computation (if the admissible order satisfies mild assumptions) shows that the theory of K-algebras is effectively noetherian over the theory of K-vector spaces;
- one can manifacture Fourier-Motzkin QE in order to show that linear rational arithmetic (with $<$) is effectively noetherian over linear rational arithmetic (without $<$);

§2. Termination by Noetherianity.

We supply here some examples (see [Nicolini, 2006] for details):

- standard Gröbner basis computation (if the admissible order satisfies mild assumptions) shows that the theory of K-algebras is effectively noetherian over the theory of K-vector spaces;
- one can manifacture Fourier-Motzkin QE in order to show that linear rational arithmetic (with $<$) is effectively noetherian over linear rational arithmetic (without $<$);
- the theory having infinitely many 0-ary predicates saying that the domain has less than n elements is noetherian and superposition calculus (in suitable cases of termination) give examples of effectively noetherian extensions of this theory;

§2. Termination by Noetherianity.

- an argument based on Dikson lemma can be used to get many-sorted versions of the latter example;

§2. Termination by Noetherianity.

- an argument based on Dikson lemma can be used to get many-sorted versions of the latter example;
- if T is stably infinite, then the extension of T with a free function symbol f is effectively noetherian over the empty theory in the signature $\{f\}$ (see [Ghilardi, Ranise, Nicolini and Zucchelli, FroCoS 07]).

§2. Termination by Noetherianity.

- an argument based on Dikson lemma can be used to get many-sorted versions of the latter example;
- if T is stably infinite, then the extension of T with a free function symbol f is effectively noetherian over the empty theory in the signature $\{f\}$ (see [Ghilardi, Ranise, Nicolini and Zucchelli, FroCoS 07]).

In all the above cases, the larger theory T is not only effectively noetherian over the smaller theory T_0, but it is also T_0-compatible, hence all such T satisfy the combinability requirements over T_0 stated in the following:

§2. Termination by Noetherianity.

$\boxed{\text{Theorem 5.}}$ [Ghilardi, Nicolini and Zucchelli, FroCoS 05] *Suppose that there is a noetherian and universal* Σ_0*-subtheory* T_0 *of* T_1 *and* T_2 *which also admits a model completion. If* T_1, T_2 *are both* T_0*-compatible and effectively noetherian extensions of* T_0*, then decidability of constraint satisfiability problem transfers from* T_1 *and* T_2 *to* $T_1 \cup T_2$*.*

§2. Termination by Noetherianity.

$\boxed{\text{Theorem 5.}}$ [Ghilardi, Nicolini and Zucchelli, FroCoS 05] *Suppose that there is a noetherian and universal* Σ_0*-subtheory* T_0 *of* T_1 *and* T_2 *which also admits a model completion. If* T_1, T_2 *are both* T_0*-compatible and effectively noetherian extensions of* T_0*, then decidability of constraint satisfiability problem transfers from* T_1 *and* T_2 *to* $T_1 \cup T_2$*.*

For earlier results on non-disjoint combined CS problems, see [Tinelli, Ringeissen 2003]. Further non-disjoint combination results arise also in connection to *locality of theory extensions* - a subject deserving a little course by itself (see [Sofronie-Stokkermans 06] for a survey).

§2. Termination by Noetherianity.

$\boxed{\text{Theorem 5.}}$ [Ghilardi, Nicolini and Zucchelli, FroCoS 05] *Suppose that there is a noetherian and universal Σ_0-subtheory T_0 of T_1 and T_2 which also admits a model completion. If T_1, T_2 are both T_0-compatible and effectively noetherian extensions of T_0, then decidability of constraint satisfiability problem transfers from T_1 and T_2 to $T_1 \cup T_2$.*

For earlier results on non-disjoint combined CS problems, see [Tinelli, Ringeissen 2003]. Further non-disjoint combination results arise also in connection to *locality of theory extensions* - a subject deserving a little course by itself (see [Sofronie-Stokkermans 06] for a survey).

The combined CS problems we analyzed so far can be generalized in various ways.

§3. Variations and Extensions.

One can consider combination constructions acting on theories different from set-theoretic union: an example of this is the *theory connection* schema analyzed in [Baader, Ghilardi 2007]: this is a special case of a cocomma category construction from categorical logic and is analogous to the E-connections known from modal/description logics.

§3. Variations and Extensions.

One can consider combination constructions acting on theories different from set-theoretic union: an example of this is the *theory connection* schema analyzed in [Baader, Ghilardi 2007]: this is a special case of a cocomma category construction from categorical logic and is analogous to the E-connections known from modal/description logics.

One can go beyond the limits of first-order logics: in [Ghilardi, Nicolini, Zucchelli 05] *higher order logic is used to specify a combination schema* that, once applied to decidable fragments of first-order languages, may produce interesting new decidable fragments (for instance, one can analyze in this way monodic fragments of first order temporal logics and recover decidability results from recent literature as instances of generalized Nelson-Oppen combination algorithms).

References:

[1] F. Baader, S. Ghilardi, *Connecting Many-Sorted Theories*, Journal of Symbolic Logic, 2007.

[2] S. Ghilardi, *Model-theoretic methods in combined constraint satisfiability*, Journal of Automated Reasoning, 2004.

[3] C. Tinelli, C. Ringeissen, *Unions of non-disjoint theories and combinations of satisfiability procedures*, Theoretical Computer Science, 2003.

[4] S. Ghilardi, E. Nicolini, and D. Zucchelli, *A comprehensive framework for combining decision procedures*, Proc. of FroCoS 05, Vol. 3717 of *LNAI*, 2005 (journal version to app. in the ACM Transactions on Computational Logic).

[5] S. Ghilardi, E. Nicolini, S. Ranise and D. Zucchelli, *Noetherianity and Combination Problems*, Proc. of FroCoS 07, *LNAI*, to app.

[6] E. Nicolini, *Combined Decision Procedures for Constraint Satisfiability*, PhD. Thesis, Università degli Studi di Milano, 2006.

[7] V. Sofronie-Stokkermans, *Local Reasoning in Verification*, Proc. of VERIFY 06, 2006.

Part III

Combined Word Problems

§1. The Problem.

In the remaining part of today slides we assume our signatures to be purely functional and our theories to be first-order **equational** theories.

§1. The Problem.

In the remaining part of today slides we assume our signatures to be purely functional and our theories to be first-order **equational** theories.

The **word problem** for such an equational Σ-theory T is to show whether a given Σ-atom $t = u$ is a logical consequence of T or not.[a]

§1. The Problem.

In the remaining part of today slides we assume our signatures to be purely functional and our theories to be first-order **equational** theories.

The **word problem** for such an equational Σ-theory T is to show whether a given Σ-atom $t = u$ is a logical consequence of T or not.[a]

The **combined word problem** for T_1, T_2 is the following: suppose that T_1, T_2 have decidable word problem, *what can we say about decidability of the word problem in* $T_1 \cup T_2$*?*

§1. The Problem.

In the remaining part of today slides we assume our signatures to be purely functional and our theories to be first-order **equational** theories.

The **word problem** for such an equational Σ-theory T is to show whether a given Σ-atom $t = u$ is a logical consequence of T or not.[a]

The **combined word problem** for T_1, T_2 is the following: suppose that T_1, T_2 have decidable word problem, *what can we say about decidability of the word problem in* $T_1 \cup T_2$*?*

Warning. One cannot directly use Nelson-Oppen schema for attacking this: purification and propagation steps produce constraints that cannot be handled by the input algorithms!

---

[a] Notice that when people in computer algebra speak of 'word problems', they deal with identity of elements in finitely presented algebras: these problems usually correspond to CS problems in our sense.

§1. The Problem.

The following result was known since long time:

§1. The Problem.

The following result was known since long time:

$\boxed{\text{Theorem 6.}}$ [Pigozzi, 1974] *If* T_1 *and* T_2 *have disjoint signatures, then decidability of word problem transfers from* T_1 *and* T_2 *to* $T_1 \cup T_2$*.*

§1. The Problem.

The following result was known since long time:

$\boxed{\text{Theorem 6.}}$ [Pigozzi, 1974] *If T_1 and T_2 have disjoint signatures, then decidability of word problem transfers from T_1 and T_2 to $T_1 \cup T_2$.*

An early attempt to drop the disjoint signature requirement was done by [Domenjoud, Klay, Ringeissen, 1994]: the results from this paper were strenghtened by the next theorem (proved independently - and with different techniques - in [Baader, Tinelli 2002] and [Fiorentini, Ghilardi 2003]).

§2. A First Combination Result.

Let T_0, T be equational theories in respective signatures Σ_0, Σ such that $\Sigma_0 \subseteq \Sigma$ and $T_0 \subseteq T$.

§2. A First Combination Result.

Let T_0, T be equational theories in respective signatures Σ_0, Σ such that $\Sigma_0 \subseteq \Sigma$ and $T_0 \subseteq T$.

$\boxed{\text{Definition}}$ *We say that T is constructible over T_0 iff T is a conservative extension of T_0 and there is a set G of Σ-terms such that:*

(i) *G contains the variables and is closed under renamings;*

(ii) *every Σ-terms factors "uniquely" (modulo T- and T_0-equivalence) as*

$$u(g_1, \ldots, g_k)$$

where u is a Σ_0-term and $\{g_1, \ldots, g_k\} \subseteq G$.

§2. A First Combination Result.

T is said to be *effectively* constructible over T_0 iff the terms $u, g_1, \ldots, g_k$ in (ii) above can be effectively computed.

§2. A First Combination Result.

T is said to be *effectively* constructible over T_0 iff the terms $u, g_1, \dots, g_k$ in (ii) above can be effectively computed.

One can get typical examples of this definition in abstract algebra (but not only there): rings are effectively constructible over abelian groups, differential rings are constructible over rings, etc.

§2. A First Combination Result.

T is said to be *effectively* constructible over T_0 iff the terms $u, g_1, \dots, g_k$ in (ii) above can be effectively computed.

One can get typical examples of this definition in abstract algebra (but not only there): rings are effectively constructible over abelian groups, differential rings are constructible over rings, etc.

$\boxed{\text{Theorem 7.}}$ [cit., 2002-03] *If T_1 and T_2 are both effectively constructible over a theory T_0 in the common subsignature, then decidability of word problem transfers from T_1 and T_2 to $T_1 \cup T_2$.*

§3. A Second Combination Result.

The assumptions of Theorem 7 are symmetric, but of syntactic nature. We give here a second recent result (Theorem 8 below) having symmetric model-theoretic assumptions.

§3. A Second Combination Result.

The assumptions of Theorem 7 are symmetric, but of syntactic nature. We give here a second recent result (Theorem 8 below) having symmetric model-theoretic assumptions.

We try to adapt Nelson-Oppen schema; besides T_0-compatibility, we need an extra assumption on the shared subtheory, whose role is that of allowing the conversion of positive constraints into rewrite rules.

§3. A Second Combination Result.

The assumptions of Theorem 7 are symmetric, but of syntactic nature. We give here a second recent result (Theorem 8 below) having symmetric model-theoretic assumptions.

We try to adapt Nelson-Oppen schema; besides T_0-compatibility, we need an extra assumption on the shared subtheory, whose role is that of allowing the conversion of positive constraints into rewrite rules.

Such an extra assumption is inspired by Gaussian elimination from linear algebra. Thus, among such 'Gaussian', theories we have the theory of vector spaces on a given field, but also some others, like - surprisingly! - Boolean algebras.

§3. A Second Combination Result.

An e-formula is a conjunctions of equations; we use $\underline{x}, \underline{y}$ for tuples of variables.

§3. A Second Combination Result.

An e-formula is a conjunctions of equations; we use $\underline{x}, \underline{y}$ for tuples of variables.

$\boxed{\text{Definition}}$ *An equational theory T_0 is* Gaussian *iff for every e-formula $\varphi(\underline{x}, y)$ it is possible to compute an e-formula $C(\underline{x})$ and a term $s(\underline{x}, \underline{z})$ with fresh variables $\underline{z}$ such that*

$$T_0 \models \varphi(\underline{x}, y) \leftrightarrow (C(\underline{x}) \wedge \exists \underline{z}.(y = s(\underline{x}, \underline{z})))$$

§3. A Second Combination Result.

An e-formula is a conjunctions of equations; we use $\underline{x}, \underline{y}$ for tuples of variables.

$\boxed{\text{Definition}}$ *An equational theory T_0 is* Gaussian *iff for every e-formula $\varphi(\underline{x}, y)$ it is possible to compute an e-formula $C(\underline{x})$ and a term $s(\underline{x}, \underline{z})$ with fresh variables $\underline{z}$ such that*

$$T_0 \models \varphi(\underline{x}, y) \leftrightarrow (C(\underline{x}) \wedge \exists \underline{z}.(y = s(\underline{x}, \underline{z})))$$

We call the formula C the *solvability condition* of φ w.r.t. y, and the term s a *(local) solver* of φ w.r.t. y in T_0.

§3. A Second Combination Result.

We are ready for our second combination result concerning word problems:

§3. A Second Combination Result.

We are ready for our second combination result concerning word problems:

$\boxed{\text{Theorem 8.}}$ [Baader, Ghilardi, Tinelli 2006] *Let T_0, T_1, T_2 be three equational theories of respective signatures $\Sigma_0, \Sigma_1, \Sigma_2$ such that*

- $\Sigma_0 = \Sigma_1 \cap \Sigma_2$;
- *T_0 is Gaussian and effectively locally finite;*
- *for $i = 1, 2$, the theory T_i is T_0-compatible and a conservative extension of T_0.*

If the word problem in T_1 and in T_2 is decidable, then the word problem in $T_1 \cup T_2$ is also decidable.

§3. A Second Combination Result.

Since the theory of Boolean algebras can be shown to be Gaussian, the following result follows, solving a long-standing open problem in modal logic:

§3. A Second Combination Result.

Since the theory of Boolean algebras can be shown to be Gaussian, the following result follows, solving a long-standing open problem in modal logic:

$\boxed{\text{Corollary 9.}}$ *If two classical multimodal logics are decidable, so is their fusion.*

§3. A Second Combination Result.

Since the theory of Boolean algebras can be shown to be Gaussian, the following result follows, solving a long-standing open problem in modal logic:

Corollary 9. *If two classical multimodal logics are decidable, so is their fusion.*

Notice that we used Segerberg's definition, according to which a modal logic is classical iff it has an algebraic semantics (i.e. classical modal logics are in bijective correspondence to equational classes of Boolean-algebras-endowed-with-further-operations). In particular, a classical modal logic needs not be normal.

§3. A Second Combination Result.

Let’s try to give some informal explanation about the use of Gaussianity in the combined algorithm of Theorem 8.

§3. A Second Combination Result.

Let's try to give some informal explanation about the use of Gaussianity in the combined algorithm of Theorem 8.

If we negate the input equation to be tested for validity and if we purify it, we get a constraint (to be tested for un-satisfiability) of the kind

$$u_1 \neq u_2 \wedge a_1 = t_1 \wedge \cdots \wedge a_n = t_n \tag{2}$$

(we use free constants like $a_1, a_2, \ldots$ instead of variables), where each of the t_j is a pure term and u_1, u_2 are Σ_0-terms.

§3. A Second Combination Result.

Let's try to give some informal explanation about the use of Gaussianity in the combined algorithm of Theorem 8.

If we negate the input equation to be tested for validity and if we purify it, we get a constraint (to be tested for un-satisfiability) of the kind

$$u_1 \neq u_2 \wedge a_1 = t_1 \wedge \cdots \wedge a_n = t_n \tag{2}$$

(we use free constants like $a_1, a_2, \ldots$ instead of variables), where each of the t_j is a pure term and u_1, u_2 are Σ_0-terms.

The equational part of (2) can be shown to consists on two pure and convergent rewrite systems $R_1 \cup R_2$ (where equations $a_j = t_j$ are oriented as ground rewrite rules $a_j \to t_j$).

§3. A Second Combination Result.

The trick is that of updating R_1, R_2 in such a way to keep them convergent and to force them to entail the same Σ_0-equations (if this task is accomplished, then it is sufficient to R_i-normalize and check for T_i-validity the equation $u_1 = u_2$, for $i = 1$ or $i = 2$ indifferently).

§3. A Second Combination Result.

The trick is that of updating R_1, R_2 in such a way to keep them convergent and to force them to entail the same Σ_0-equations (if this task is accomplished, then it is sufficient to R_i-normalize and check for T_i-validity the equation $u_1 = u_2$, for $i = 1$ or $i = 2$ indifferently).

In fact, this updating is done by induction on any strict total order on the constants 'compatible' with the rewriting system $R_1 \cup R_2$.

§3. A Second Combination Result.

The trick is that of updating R_1, R_2 in such a way to keep them convergent and to force them to entail the same Σ_0-equations (if this task is accomplished, then it is sufficient to R_i-normalize and check for T_i-validity the equation $u_1 = u_2$, for $i = 1$ or $i = 2$ indifferently).

In fact, this updating is done by induction on any strict total order on the constants 'compatible' with the rewriting system $R_1 \cup R_2$.

When examining $a_j \to t_j$ (suppose t_j is a Σ_1-term) one collects all representative $\Sigma_0(a_1, \ldots, a_j)$-equations that normalize to a T_1-valid equation using R_1. The conjunctions of these equations is then solved with respect to a_j and the skolemized solver s gives a new rewrite rule $a_j \to s$ to be added to R_2 in order to update it.

§3. A Second Combination Result.

Theorem 6 and Theorem 7 both imply the disjoint signature case covered by Theorem 5, but none of them is stronger than the other.

§3. A Second Combination Result.

Theorem 6 and Theorem 7 both imply the disjoint signature case covered by Theorem 5, but none of them is stronger than the other.

Whereas it is well-known that combined word problems may be undecidable in the non-disjoint signature case, we tend to believe that substantial work can still be done in this area.

References:

[1] F. Baader, S. Ghilardi, C. Tinelli, *A new combination procedure for the word problem that generalizes fusion decidability results in modal logics*, Information and Computation, 2006.

[2] F. Baader, C. Tinelli, *Deciding the word problem in the union of equational theories*, Information and Computation, 2002.

[3] E. Domenjoud, F. Klay, C. Ringeissen, *Combination techniques for non-disjoint equational theories*, Proc. of CADE 12, Vol. 814 of *LNAI*, 1994.

[4] C. Fiorentini, S. Ghilardi, *Combining word problems through rewriting in categories with products*, Theoretical Computer Science, 2003.

[5] D. Pigozzi, *The join of equational theories*, Colloquium Mathematicum, 1974.