Formale Methoden der Informatik

Block 2: Satisfiability Problems

5. Equality Logic and Uninterpreted Function Symbols

Uwe Egly

Knowledge-Based Systems Group
Institute of Information Systems
Vienna University of Technology

The overall procedure again

Goal of the SAT part

Provide necessary tools and background info to construct a decision procedure for equality logic with uninterpreted functions (EUF).

Overall procedure

The problem of deciding a EUF-formula φ^{EUF} is reduced to the SAT problem of a propositional formula φ^{P} such that

$$\varphi^{EUF} \text{ is } E\text{-valid} \quad \text{iff} \quad \varphi^{E} \text{ is } E\text{-valid} \quad \text{iff} \quad \varphi^{P} \text{ is unsat}$$

Then a model of φ^{P} provides a counterexample to the E-validity of φ^{EUF}!

The task for today

- Given an input formula φ^{EUF}. We want to construct an E-formula φ^{E} such that

$$\boxed{\varphi^{EUF} \text{ is } E\text{-valid} \quad \text{if and only if} \quad \varphi^{E} \text{ is } E\text{-valid}}$$

- The construction of φ^{E} from φ^{EUF} (reduction) requires run time polynomial in the size of φ^{EUF}.

- An overview of the translation can be found in:
  D. Kroening, O. Strichman. Decision Procedures, Springer, 2008

Outline

Equality logic with uninterpreted function symbols

Reduction of uninterpreted functions to equality logic

- Ackermann's reduction

Learning Objectives

Interpreted functions

- Every function is a mapping from a domain to a range.
- Example: $+$ over $\mathbb{N}_0$ is a mapping from $\mathbb{N}_0 \times \mathbb{N}_0$ to $\mathbb{N}_0$.
- For interpreted functions, theory reasoning is involved,
  - e.g., Presburger arithmetic [link] or Peano arithmetic [link]
- What if this reasoning is computationally too demanding for (practical) use?

☞ Try to make reasoning simpler by dumping the theory underlying the function!

Uninterpreted functions (UFs)

Basic idea

Neglect basic properties of a function (except being a function).

Example

Consider $\varphi\colon x + 0 \doteq x$ which is valid in Presburger arithmetic. We replace the interpreted binary function $+$ by an uninterpreted binary function F resulting in the formula $\varphi'\colon F(x, 0) \doteq x$. Contrary to φ, φ' is not valid. The binary function symbol F can represent any binary function. It is not restricted by any theory axiom and the only property of F is: it is a function.

Uninterpreted functions cont'd

- Uninterpreted functions can be used for abstracting away certain properties and for generalizing theorems.

- The introduction of uninterpreted functions into φ makes the resulting formula φ' weaker:

If φ' is $\mathcal{T}$-valid then φ is $\mathcal{T}$-valid.

Why?

Functional consistency

- The basic axiom for any function is functional consistency. (functional congruence, substitution axiom schemes for functions)
- Meaning: If you put the same argument(s) to a function, then the same function value is returned.
- The functional consistency axiom schema is

$$x_1 \doteq x_1' \wedge \cdots \wedge x_n \doteq x_n' \rightarrow F(x_1, \ldots, x_n) \doteq F(x_1', \ldots, x_n')$$

- Sometimes functional consistency is all what we need for a proof. (The hope is that the use of UFs simplifies the proof!)
- Applications and limitations will be discussed later.

The syntax of equality logic with UFs (*EUF*)

Definition (The syntax of equality logic with UFs (*EUF*-logic))

$$
\begin{array}{rcl}
\mathit{formula} & ::= & \mathit{atom} \mid (\mathit{formula}) \mid \neg \mathit{formula} \mid \mathit{formula} \wedge \mathit{formula} \mid \\
 & & \mathit{formula} \vee \mathit{formula} \mid \mathit{formula} \rightarrow \mathit{formula} \\
\mathit{atom} & ::= & \mathit{term} \doteq \mathit{term} \mid \mathit{predicate\text{-}symbol}(\mathit{list\ of\ terms}) \\
\mathit{term} & ::= & \mathit{identifier} \mid \mathit{function\text{-}symbol}(\mathit{list\ of\ terms})
\end{array}
$$

Remarks

1. PSs (except $\doteq$) and function symbols are uninterpreted.
2. Uninterpreted function symbols are denoted by capital letters.
3. *list of terms*: a possibly empty list of terms
   ($F()$ is often identified with F)
4. Identifiers are often called (term) variables.
5. φ^{EUF} denotes an *EUF*-formula.

The removal of uninterpreted predicates (UPs)

Definition (Removal of UPs from an *EUF*-formula φ^{EUF})

Replace in φ^{EUF} any occurrence of the UP of the form $p(t_1, \ldots, t_n)$ by $F_p(t_1, \ldots, t_n) \doteq x_p$, where F_p is the new UF symbol for p and x_p is the new term variable for p.

- The resulting formula is ψ^{EUF} and does not contain any UP symbol. We have:

$$\varphi^{EUF} \text{ is } E\text{-satisfiable} \quad \text{iff} \quad \psi^{EUF} \text{ is } E\text{-satisfiable}$$

- From now on, we consider only *EUF*-formulas without UPs.

A motivating example for equality logic with UFs

```
int power3 (int in) {
  int i, out_a;
  out_a = in;
  for(i=0; i<2; i++)
    out_a = out_a * in;
  return out_a;
}
```

```
int power3_n (int in) {
  int out_b;
  out_b = (in * in) * in;
  return out_b;
}
```

- Prove equivalence of the two programs (i.e., they return, for every identical input, exactly the same value).
- Proving equivalence of programs is undecidable in general!
- It is possible here because the loop is bounded.

Compute the programs' input/output relation (1)

1. Remove variable declarations and `return` statements.
2. Unroll the `for` loop.
3. Replace each left-hand side variable in each assignment by a new auxiliary variable and ...
4. propagate the change through the program.
5. Conjoin all program statements.

Resulting formulas: φ for `power3` and φ_n for `power3_n`

Compute the programs' input/output relation (2)

```
int power3 (int in) {
  int i, out_a;
  out_a = in;
  for(i=0; i<2; i++)
    out_a = out_a * in;
  return out_a;
}
```

```
int power3_n (int in) {
  int out_b;
  out_b = (in * in) * in;
  return out_b;
}
```

φ : $o0a \doteq in \quad \wedge$
$o1a \doteq o0a * in \quad \wedge$
$o2a \doteq o1a * in$

φ_n : $o0b \doteq (in * in) * in$

We have to show the validity of $\varphi \wedge \varphi_n \rightarrow o2a \doteq o0b$!

The introduction of UFs

Problem

The proof of validity has to take machine arithmetic into account. Reasoning about 32 or 64 bit machine arithmetic is hard.

Idea

Skip nasty details by replacing multiplication by an "abstract function" G satisfying less properties.

- We obtain from φ and φ_n the formulas

$$\varphi^{EUF}: \quad o0a \doteq in \wedge o1a \doteq G(o0a, in) \wedge o2a \doteq G(o1a, in)$$
$$\varphi_n^{EUF}: \quad o0b \doteq G(G(in, in), in)$$

- Attempt to validate $\varphi^{EUF} \wedge \varphi_n^{EUF} \rightarrow o2a \doteq o0b$
  - ☞ See extra-sheet5-1.pdf in TUWEL for a detailed proof!
- If successful, then $\varphi \wedge \varphi_n \rightarrow o2a \doteq o0b$ is also valid! (The converse does not hold in general!)

Outline

How to prove a *EUF*-formula valid?

- We want to know whether a *EUF*-formula is valid.
- Instead of developing a separate algorithm for validity, we reduce it to equality logic in a validity-preserving way.
- There are two main approaches for such a reduction:
  - Ackermann's reduction (AR)
  - Bryant's reduction (BR) (not discussed in the lecture)
- We discuss the first approach in detail.

Algorithms for both can be found on p. 67 and p. 70 in D. Kroening, O. Strichman. Decision Procedures. A slightly modified version is available in TUWEL as an additional hand-out.

The key idea of the reductions

The main steps in the construction

1. Represent the propositional structure of the input EUF-formula φ^{EUF} by an E-formula $\mathit{flat}^E(\varphi^{EUF})$.
2. Construct an E-formula $FC^E(\varphi^{EUF})$ which re-introduces the effects of functional constraints.
3. The resulting E-formula φ^E is $FC^E(\varphi^{EUF}) \rightarrow \mathit{flat}^E(\varphi^{EUF})$.

Q: For which FS do we need functional constraints?

- ☞ For all FS with arity > 0! (Not for constant symbols)
- ☞ Define $\mathcal{F}(\varphi^{EUF})$: the set of all FS from φ^{EUF} with arity > 0!

The basic principle of Ackermann's reduction (1)

Given a formula φ^{EUF} with m_F instances $F_1, \ldots, F_{m_F}$ of an UF F (generalizations to more FSs are discussed later)

1. Number the function instances (e.g., from inside out):
$$F(F(x)) \doteq 0 \quad \Longrightarrow \quad F_2(F_1(x)) \doteq 0$$
2. Associate to each function instance F_i a new term variable f_i:
$$F_2(F_1(x)) \doteq 0 \quad \Longrightarrow \quad \underbrace{F_2(\overbrace{F_1(x)}^{f_1})}_{f_2} \doteq 0$$

Remarks:

- A constant or a variable is always associated to itself.
- Different occurrences of the same instance are associated with the same term variable.
- We use $\mathcal{T}(t)$ to denote the symbol associated to t.

The basic principle of Ackermann's reduction (2)

3. Compute $flat^E(\varphi^{EUF})$ by replacing top-level instances F_i by the associated term variable f_i.

$$\underbrace{F_2(\overbrace{F_1(x)}^{f_1})}_{f_2} \doteq 0 \quad \Longrightarrow \quad f_2 \doteq 0$$

4. Let $arg(F_i)$ denote the argument(s) of a function instance F_i. Compute, for every FS $F \in \mathcal{F}(\varphi^{EUF})$, the formula $FC^E_F(\varphi^{EUF})$

$$\bigwedge_{i=1}^{m_F-1} \bigwedge_{j=i+1}^{m_F} (\mathcal{T}(arg(F_i)) \doteq \mathcal{T}(arg(F_j)) \rightarrow f_i \doteq f_j).$$

Then the result is $FC^E(\varphi^{EUF})$: $\bigwedge_{F \in \mathcal{F}(\varphi^{EUF})} FC^E_F(\varphi^{EUF})$.

☞ For the example formula, we get $FC^E(\varphi^{EUF})$: $x \doteq f_1 \rightarrow f_1 \doteq f_2$.

☞ Then φ^E: $(x \doteq f_1 \rightarrow f_1 \doteq f_2) \rightarrow f_2 \doteq 0$.

Ackermann's reduction: An example

- Consider $\varphi^{EUF}: x_1 \mathrel{\dot{\neq}} x_2 \vee F(x_1) \doteq F(x_2) \vee F(x_1) \mathrel{\dot{\neq}} F(x_3)$.
- First we number the instances of the only (proper) UF (different occurrences of the same instance get the same number).

$$x_1 \mathrel{\dot{\neq}} x_2 \vee F_1(x_1) \doteq F_2(x_2) \vee F_1(x_1) \mathrel{\dot{\neq}} F_3(x_3)$$

- Compute $\mathit{flat}^E(\varphi^{EUF})$ by replacing top-level instances F_i by the associated term variable f_i.

$$x_1 \mathrel{\dot{\neq}} x_2 \vee f_1 \doteq f_2 \vee f_1 \mathrel{\dot{\neq}} f_3$$

- Add functionality constraints for F, i.e., compute $FC_F^E(\varphi^{EUF})$:

$$(x_1 \doteq x_2 \rightarrow f_1 \doteq f_2) \wedge (x_1 \doteq x_3 \rightarrow f_1 \doteq f_3) \wedge (x_2 \doteq x_3 \rightarrow f_2 \doteq f_3)$$

- With $FC_F^E(\varphi^{EUF}) = FC^E(\varphi^{EUF})$, we have

$$\varphi^E: \quad FC_F^E(\varphi^{EUF}) \rightarrow (x_1 \mathrel{\dot{\neq}} x_2 \vee f_1 \doteq f_2 \vee f_1 \mathrel{\dot{\neq}} f_3)$$

Ackermann's reduction: An example with more UFs (1)

- Consider $\varphi^{EUF} : x_1 \doteq x_2 \rightarrow F(F(G(x_1))) \doteq F(F(G(x_2)))$.
- Number the instances of the UFs separately for each FS and associate term variables.

$$x_1 \doteq x_2 \rightarrow \underbrace{F_2(\underbrace{F_1(\overbrace{G_1(x_1)}^{g_1})}_{f_1})}_{f_2} \doteq \underbrace{F_4(\underbrace{F_3(\overbrace{G_2(x_2)}^{g_2})}_{f_3})}_{f_4}$$

- Compute $flat^E(\varphi^{EUF})$ by replacing top-level instances of functions by the associated term variables resulting in

$$x_1 \doteq x_2 \rightarrow f_2 \doteq f_4.$$

Ackermann's reduction: An example with more UFs (2)

- Add functionality constraints, i.e., compute $FC_F^E(\varphi^{EUF})$ and $FC_G^E(\varphi^{EUF})$ independently and conjoin the results:

$$\begin{array}{rcll} g_1 \doteq f_1 & \rightarrow & f_1 \doteq f_2 & \wedge \\ g_1 \doteq g_2 & \rightarrow & f_1 \doteq f_3 & \wedge \\ g_1 \doteq f_3 & \rightarrow & f_1 \doteq f_4 & \wedge \\ f_1 \doteq g_2 & \rightarrow & f_2 \doteq f_3 & \wedge \\ f_1 \doteq f_3 & \rightarrow & f_2 \doteq f_4 & \wedge \\ g_2 \doteq f_3 & \rightarrow & f_3 \doteq f_4 & \wedge \\ x_1 \doteq x_2 & \rightarrow & g_1 \doteq g_2 & \end{array}$$

- Then we have φ^E:

$$FC_F^E(\varphi^{EUF}) \wedge FC_G^E(\varphi^{EUF}) \rightarrow (x_1 \doteq x_2 \rightarrow f_2 \doteq f_4)$$

The program equivalence example again (1)

- Apply AR to ψ^{EUF} : $\varphi^{EUF} \wedge \varphi_n^{EUF} \rightarrow o2a \doteq o0b$ with

$$\varphi^{EUF}: \quad o0a \doteq in \wedge o1a \doteq G(o0a, in) \wedge o2a \doteq G(o1a, in)$$
$$\varphi_n^{EUF}: \quad o0b \doteq G(G(in, in), in)$$

- Number the occurrences of G starting with 1 in the order

$$G(o0a, in), \quad G(o1a, in), \quad G(in, in), \quad G(G(in, in), in).$$

- We compute $flat^E(\psi^{EUF})$ which is

$$o0a \doteq in \wedge o1a \doteq g_1 \wedge o2a \doteq g_2 \wedge o0b \doteq g_4 \rightarrow o2a \doteq o0b.$$

The program equivalence example again (2)

Problem

So far, we dealt only with unary function symbols. What if the arity n is bigger than 1?

Solution

Simply read

$$\begin{array}{rcll} \mathcal{T}(arg(F_i)) \doteq \mathcal{T}(arg(F_j)) & \rightarrow & f_i \doteq f_j & \text{which is} \\ \mathcal{T}((s_1, \ldots, s_n)) \doteq \mathcal{T}((t_1, \ldots, t_n))) & \rightarrow & f_i \doteq f_j & \text{as} \\ (\mathcal{T}(s_1) \doteq \mathcal{T}(t_1) \wedge \cdots \wedge \mathcal{T}(s_n) \doteq \mathcal{T}(t_n)) & \rightarrow & f_i \doteq f_j & \end{array}$$

The program equivalence example again (3)

- We compute $FC_G^E(\psi^{EUF})$ which is

$$\begin{array}{rcrcrcl}
(o0a \doteq & o1a & \wedge & in \doteq in & \rightarrow & g_1 \doteq g_2) & \wedge \\
(o0a \doteq & in & \wedge & in \doteq in & \rightarrow & g_1 \doteq g_3) & \wedge \\
(o0a \doteq & g_3 & \wedge & in \doteq in & \rightarrow & g_1 \doteq g_4) & \wedge \\
(o1a \doteq & in & \wedge & in \doteq in & \rightarrow & g_2 \doteq g_3) & \wedge \\
(o1a \doteq & g_3 & \wedge & in \doteq in & \rightarrow & g_2 \doteq g_4) & \wedge \\
(in \doteq & g_3 & \wedge & in \doteq in & \rightarrow & g_3 \doteq g_4) &
\end{array}$$

- Remark: $FC_G^E(\psi^{EUF})$ can be simplified!
- We get $\varphi^E : FC_G^E(\psi^{EUF}) \rightarrow flat^E(\psi^{EUF})$

How can we decide φ^E?

Homework

Show the following:

φ^{EUF} is satisfiable iff $FC^E(\varphi^{EUF}) \wedge flat^E(\varphi^{EUF})$ is satisfiable.

$FC^E(\varphi^{EUF})$ and $flat^E(\varphi^{EUF})$ are obtained from φ^{EUF} by Ackermann's reduction.

(Hint: $FC^E(\varphi^{EUF}) = FC^E(\neg\varphi^{EUF})$)

The big picture again

1. Given a formula φ^{EUF}: Try to prove it E-valid.

The big picture again

1. Given a formula φ^{EUF}: Try to prove it E-valid.
2. Translate it to φ^{E} by AR (it preserves E-validity).

The big picture again

1. Given a formula φ^{EUF}: Try to prove it E-valid.
2. Translate it to φ^{E} by AR (it preserves E-validity).
3. φ^{E} : $FC^{E}(\varphi^{EUF}) \rightarrow flat^{E}(\varphi^{EUF})$ and φ^{EUF} is E-valid iff φ^{E} is E-valid.

The big picture again

1. Given a formula φ^{EUF}: Try to prove it E-valid.
2. Translate it to φ^{E} by AR (it preserves E-validity).
3. $\varphi^{E}: FC^{E}(\varphi^{EUF}) \rightarrow flat^{E}(\varphi^{EUF})$ and φ^{EUF} is E-valid iff φ^{E} is E-valid.
4. Translate to SAT:
   φ^{EUF} is E-valid iff $\varphi_1^{E}: FC^{E}(\varphi^{EUF}) \wedge \neg flat^{E}(\varphi^{EUF})$ is E-unsatisfiable

The big picture again

1. Given a formula φ^{EUF}: Try to prove it E-valid.
2. Translate it to φ^{E} by AR (it preserves E-validity).
3. $\varphi^{E}: FC^{E}(\varphi^{EUF}) \rightarrow flat^{E}(\varphi^{EUF})$ and φ^{EUF} is E-valid iff φ^{E} is E-valid.
4. Translate to SAT:
   φ^{EUF} is E-valid iff $\varphi_1^{E}: FC^{E}(\varphi^{EUF}) \wedge \neg flat^{E}(\varphi^{EUF})$ is E-unsatisfiable

$$
\begin{array}{lll}
\varphi^{EUF}\ E\text{-valid} & \text{iff} & \varphi^{E}\ E\text{-valid} \\
 & \text{iff} & \neg\varphi^{E}\ E\text{-unsatisfiable} \\
 & \text{iff} & \neg\left(FC^{E}(\varphi^{EUF}) \rightarrow flat^{E}(\varphi^{EUF})\right)\ E\text{-unsatisfiable} \\
 & \text{iff} & FC^{E}(\varphi^{EUF}) \wedge \neg flat^{E}(\varphi^{EUF})\ E\text{-unsatisfiable}
\end{array}
$$

The big picture again

1. Given a formula φ^{EUF}: Try to prove it E-valid.
2. Translate it to φ^{E} by AR (it preserves E-validity).
3. φ^{E} : $FC^{E}(\varphi^{EUF}) \rightarrow flat^{E}(\varphi^{EUF})$ and φ^{EUF} is E-valid iff φ^{E} is E-valid.
4. Translate to SAT:
   φ^{EUF} is E-valid iff φ_1^{E} : $FC^{E}(\varphi^{EUF}) \wedge \neg flat^{E}(\varphi^{EUF})$ is E-unsatisfiable
5. Simplification yields φ_2^{E} with φ_2^{E} is E-sat iff φ_1^{E} is E-sat.

   Therefore φ^{EUF} is E-valid iff φ_2^{E} is E-unsatisfiable.

The big picture again (cont'd)

6. Generate from φ_2^E the propositional formula φ^P with φ^P is satisfiable iff φ_2^E is E-satisfiable.
   Therefore φ^{EUF} is E-valid iff φ^P is unsatisfiable.

The big picture again (cont'd)

6. Generate from φ_2^E the propositional formula φ^P with
   φ^P is satisfiable iff φ_2^E is E-satisfiable.
   Therefore φ^{EUF} is E-valid iff φ^P is unsatisfiable.

7. Give φ^P to a SAT solver. Possible results are:

UNSAT Then φ^{EUF} is E-valid.

SAT Then φ^{EUF} is not E-valid. A counter-example to the E-validity of φ^{EUF} can be constructed from a model of φ^P.

Learning objectives

You should be able to

- explain the syntax of equality logic with UFs,
- show (using semantics) an *EUF*-formula valid,
- explain how uninterpreted predicates can be eliminated from an *EUF*-formula (including the correctness proof),
- explain the essence of uninterpreted functions,
- translate an *EUF*-formula to an E-formula,
- define AR for more than one FS and for FSs with arity > 1,
- explain in detail and prove the logical connections (e.g., wrt validity and satisfiability) between φ^{EUF}, φ^{E} and φ^{P}.