Non-classical logics

Lecture 9 + 10:

– Many-valued logics: Applications in verification

– Infinitely-valued logics

4.12.2013

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Until now

- Many-valued logic (finitely-valued)

  History and Motivation
  Syntax /Semantics

  Functional completeness

  Automated reasoning: Tableaux, Resolution

  Applications: logic

Applications of many-valued logic

- independence proofs
- modeling undefined function and predicate values (program verification)
- semantic of natural languages
- theory of logic programming: declarative description of operational semantics of negation
- modeling of electronic circuits
- modeling vagueness and uncertainly
- shape analysis (program verification)

Applications of many-valued logic

- independence proofs last time
- modeling undefined function and predicate values (program verification)
- semantic of natural languages
- theory of logic programming: declarative description of operational semantics of negation
- modeling of electronic circuits
- modeling vagueness and uncertainly
- shape analysis (program verification) today

Shape analysis

Shape Analysis is an important and well covered part of static program analysis.

The central role in shape analysis is played by the set U of abstract stores.

U is perceived as the abstraction of the locations program variables can point to.

In an object-oriented context U can be viewed as an abstraction of the set of all objects existing at a snapshot during program execution

Shape analysis

U set of abstract stores.

X set of program variables.

Abstract state of a program at a given snapshot:

- Structure $\mathcal{S} = (U, \{x : U \to \{0,1\}\}_{x \in X} \cup$ Additional predicates)

  $x(v) = 1$ (also denoted $\mathcal{S} \models x[v]$) iff variable x points to store v.

For any abstract state $\mathcal{S}$ and any program variable x we require that the unary predicate x holds true of at most one store, i.e. we require

$$\mathcal{S} \models \forall s_1 \forall s_2 ((x(s_1) \wedge x(s_2)) \to s_1 = s_2).$$

It is possible that x does not point to any store, i.e. $\mathcal{S} \models \forall s (\neg x(s))$.

Shape analysis

Additional predicates on $\mathcal{S}$ depend on the specific program/task

Example: next : $U^2 \to \{0, 1\}$

Examples of properties:

$\exists s\ x(s)$	x does not point to null
$\forall s(\neg(x(s) \wedge t(s)))$	x and t do not point to the same store
$\exists s\ \mathrm{is}(s)$	the list defined by next contains a shared node

We have used the abbreviation

$$\mathrm{is}(s) = \exists s_1 \exists s_2 (next(s_1, s) \wedge next(s_2, s) \wedge s_1 \neq s_2)$$

Goal: prove for a given program, or a given program part, that a certain property holds at every program state, or every stable program state.

Example: List reversing

Goal: Cycle-freeness of a list pointer structure is preserved by the algorithm reversing the list.

Describing cycle-freeness

1. $\neg\exists v(next(v, n)$ n is the store representing the head of the list
2. $\forall v \forall w(next(m, v) \wedge next(m, w) \rightarrow v = w)$ for all stores m reachable from n,
3. $\neg \mathrm{is}(m)$ for all stores m reachable from n.

Remark:
If conditions 1.–3. hold then the list with entry point n cannot be cyclic.

We concentrate here on showing the preservation of the formula $\mathrm{is}(s)$.

Example: List reversing

Algorithm for list reversing:

```
class ReverseList {
   int value;
   ReverseList next;


public ReverseList reverse() {
       ReverseList t, y= null, x = this;
       while (x != null) {
       st1: t=y;
       st2: y=x;
       st3: x=x.next;
       st4: y.next = t;}
       return y;}}
```

Example: List reversing

Task:

Assume that at the beginning of the while loop $\mathcal{S} \models \neg is(n)$ is true for all stores n in the list.

Show that in the state $\mathcal{S}_e$ after execution of the while loop again $\mathcal{S}_e \models \neg is(n)$ holds true for all n.

Problem: Since we cannot make any assumptions on the set of stores U at the start of the while-loop we need to investigate infinitely many structures, which obviously is not possible.

Shape analysis

Idea [Mooly Sagiv, Thomas Reps and Reinhard Wilhelm]

Use of three-valued structures to approximate two-valued structures.

More precisely, we try to find finitely many three-valued structures $\mathcal{S}_1^3, \ldots, \mathcal{S}_k^3$ such that for an arbitrary two-valued abstract state $\mathcal{S}$ that may be possible before the while-loop starts there is a surjective mapping F from S onto one of the $\mathcal{S}_i^3$ for $1 \leq i \leq k$ with $\mathcal{S} \sqsubseteq^F \mathcal{S}_i^3$, i.e.

- for all n-ary predicate symbols p and all $b_1, \ldots, b_n \in U_\mathcal{S}$ we have:

$$p_{\mathcal{S}_i^3}(F(b_1), \ldots, F(b_n)) \leq_i p_\mathcal{S}(b_1, \ldots, b_n)$$

bb where $a \leq_i b$ iff $a = b$ or $a = \frac{1}{2}$

(every possible initial state has an abstraction among $\mathcal{S}_1^3, \ldots, \mathcal{S}_k^3$)

Shape analysis

Plan:

Step 1:

For every three-valued structure $\mathcal{S}_i^3$ we will define an algorithm to compute a three-valued structure $\mathcal{S}_{i,e}^3$.

We think of $\mathcal{S}_{i,e}^3$ as the three-valued state reached after execution of α_r (the body of the while-loop) when started in $\mathcal{S}_i^3$.

If $\mathcal{S}$ is a two-valued state it is fairly straight forward to compute the two-valued state $\mathcal{S}_e$ that is reached after executing α_r starting with $\mathcal{S}$, since the commands in α_r are so simple.

The construction of $\mathcal{S}_{i,e}^3$ will be done such that $\mathcal{S} \sqsubseteq^F \mathcal{S}_i^3$ implies $\mathcal{S}_e \sqsubseteq^F \mathcal{S}_{i,e}^3$.

Shape analysis

Plan:

Step 2:

Determine a set $\mathcal{M}_0$ of abstract three-valued states to start with.

Shape analysis

Plan:

Step 3:

At iteration $k (k \geq 1)$ we are dealing with a set $\mathcal{M}_{k-1}$ of abstract three-valued states.

We try to prove for every $\mathcal{S}^3 \in \mathcal{M}_{k-1}$ that if $\mathcal{S}^3 \models \forall s(\neg \mathrm{is}(s)))$ then $\mathcal{S}^3_e \models (\forall s(\neg \mathrm{is}(s)))$.

It will then follow that for any two-valued state $\mathcal{S}$ that is reachable with $k - 1$ iterations of α_r:

$$\mathcal{S} \models \forall \neg \mathrm{is}(s) \Rightarrow \mathcal{S}_e \models \forall s \neg \mathrm{is}(s)$$

If we succeed we set

$$\mathcal{M}_k = \{\mathcal{S}^3_e | \mathcal{S}^3 \in \mathcal{M}_{k-1}\}$$

Shape analysis

Plan:

Step 3 (continued)

If $\mathcal{M}_k \subseteq \mathcal{M}_{k-1}$ we are finished and the claim is positively established.

Otherwise we repeat step 3 with $\mathcal{M}_k$.

If for one $\mathcal{S}^3 \in \mathcal{M}_{k-1}$, $\forall s(\neg \mathrm{is}(s)))$ evaluated to 0 then our conjecture was false.

If for one $\mathcal{S}^3 \in \mathcal{M}_{k-1}$, $\forall s(\neg \mathrm{is}(s)))$ evaluated to $\frac{1}{2}$ then this result is inconclusive. Should this happen we need to iterate the procedure with a larger set $\mathcal{M}'_{k-1}$.

There is, unfortunately, no guarantee that this iteration will come to a conclusive end in the general case.

Shape analysis

[Example on the blackboard]

cf. also P.H. Schmidt's lecture notes, Section 2.4.4 (pages 91-100).

Conclusions

- Finitely-valued logics: natural generalization of classical logic

- Tableau calculi
- Resolution

  extend in a natural way

- Applications

Similar results also for logics with infinitely many truth values?

Infinitely-Valued Logics

Łukasiewicz logics

Łukasiewicz logics

$\mathcal{L}_n,\ n \in \mathbb{N}$	$W_n = \{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}$
$\mathcal{L}_{\aleph_0}$	$W_{\aleph_0} = [0,1] \cap \mathbb{Q}$
$\mathcal{L}_{\aleph_1}$	$W_{\aleph_1} = [0,1]$

Logical operations: $\vee, \wedge, \neg, \Rightarrow$

- $\vee = \max$
- $\wedge = \min$
- $\neg x = 1 - x$
- $x \Rightarrow y = \min(1, 1 - x + y)$

Łukasiewicz logics

Łukasiewicz implication $x \Rightarrow_{Ł_n} y = \min(1, 1 - x + y)$

$\mathcal{L}_n$

$\Rightarrow$	0	$\frac{1}{n-1}$	$\frac{2}{n-1}$	$\dots$	$\frac{n-2}{n-1}$	1
0	1	1	1	$\dots$	1	1
$\frac{1}{n-1}$	$\frac{n-2}{n-1}$	1	1	$\dots$	1	1
$\frac{2}{n-1}$	$\frac{n-3}{n-1}$	$\frac{n-2}{n-1}$	1	$\dots$	1	1
$\dots$						
1	0	$\frac{1}{n-1}$	$\frac{2}{n-1}$	$\dots$	$\frac{n-2}{n-1}$	1

Łukasiewicz logics

Theorems.

1. For $n, m \in \mathbb{N}$, s.t. $(m-1)|(n-1)$, we have Tautologies($\mathcal{L}_n$) $\subseteq$ Tautologies($\mathcal{L}_m$)
2. Tautologies($\mathcal{L}_{\aleph_0}$) $=$ Tautologies($\mathcal{L}_{\aleph_1}$)
3. Tautologies($\mathcal{L}_{\aleph_0}$) $= \bigcap\{$Tautologies($\mathcal{L}_n$) $\mid n \geq 2, n \in \mathbb{N}\}$

Proofs

Theorem. For $n, m \in \mathbb{N}$, s.t. $(m-1)|(n-1)$, we have Tautologies$(\mathcal{L}_n) \subseteq$ Tautologies$(\mathcal{L}_m)$

Proof

Assume $(m-1)|(n-1)$. Then $W_m \subseteq W_n$. Assume $F \in$ Tautologies$(\mathcal{L}_n)$. Then F evaluates to 1 under every valuation into W_n, hence also under every valuation into W_m, so $F \in$ Tautologies$(\mathcal{L}_m)$

Proofs

Theorem. For $n, m \in \mathbb{N}$, s.t. $(m-1)|(n-1)$, we have Tautologies($\mathcal{L}_n$) $\subseteq$ Tautologies($\mathcal{L}_m$)

Remark: the converse also holds

If Tautologies($\mathcal{L}_n$) $\subseteq$ Tautologies($\mathcal{L}_m$) then $(m-1)|(n-1)$.

(This will be discussed in the next exercise session.)

Proofs

Theorem.
$\text{Tautologies}(\mathcal{L}_{\aleph_0}) = \text{Tautologies}(\mathcal{L}_{\aleph_1})$

Proof.
" $\supseteq$ " : Since $[0,1] \cap \mathbb{Q} \subseteq [0,1]$, it is clear that
$\text{Tautologies}(\mathcal{L}_{\aleph_1}) \subseteq \text{Tautologies}(\mathcal{L}_{\aleph_0})$

Proofs

Theorem.
Tautologies($\mathcal{L}_{\aleph_0}$) $=$ Tautologies($\mathcal{L}_{\aleph_1}$)

Proof.
" $\subseteq$ " : Let $F \in$ Tautologies($\mathcal{L}_{\aleph_0}$). Then for every assignment of values in $[0,1] \cap \mathbb{Q}$ to the propositional variables $\{P_1, \dots, P_n\}$ of F evaluates to 1.

We can associate a function $f_F : [0,1]^n \to [0,1]$ with F which is defined as follows:
For all $(x_1, \dots, x_n) \in [0,1]^n$ let $\mathcal{A} : \{P_1, \dots, P_n\} \to [0,1]$ be defined by $\mathcal{A}(P_i) = x_i$.
We define $f_F(x_1, \dots, x_n) := \mathcal{A}(F)$

It can be proved by structural induction that f_F is a continuous function.

Let $(a_1, \dots, a_n) \in [0,1]^n$. It is now sufficient to choose sequences of rational numbers converging to $a_1, \dots, a_n$ respectively. $f_F(a_1, \dots, a_n)$ is the limit of the sequence defined this way, hence its value is 1.

Proofs

$\text{Tautologies}(\mathcal{L}_{\aleph_0}) = \bigcap\{\text{Tautologies}(\mathcal{L}_n) \mid n \geq 2, n \in \mathbb{N}\}$

Proof.

" $\subseteq$ " : Follows from the fact that $W_n \subseteq [0, 1] \cap \mathbb{Q}$ for every $n \in \mathbb{N}$.

Proofs

$\text{Tautologies}(\mathcal{L}_{\aleph_0}) = \bigcap\{\text{Tautologies}(\mathcal{L}_n) \mid n \geq 2, n \in \mathbb{N}\}$

Proof. " $\supseteq$ "

Let F be a formula with prop. variables $\{P_1, \ldots, P_k\}$ s.t. $F \notin \text{Tautologies}(\mathcal{L}_{\aleph_0})$. Then there exists $\mathcal{A} : \{P_1, \ldots, P_k\} \rightarrow [0,1] \cap \mathbb{Q}$ s.t. $\mathcal{A}(F) \neq 1$.

Assume that $\mathcal{A}(P_1) = \frac{q_1}{p_1}, \ldots, \mathcal{A}(P_k) = \frac{q_k}{p_k}$

Let $m = \text{lcm}(p_1, \ldots, p_k)$. Then it is easy to see that $\mathcal{A}(P_i) \in W_{m+1}$ for all $1 \leq i \leq k$.

We thus constructed a valuation $\mathcal{A} : \{P_1, \ldots, P_k\} \rightarrow W_m$ such that $\mathcal{A}(F) \neq 1$. Hence, $F \notin \text{Tautologies}(\mathcal{L}_m)$, so

$$F \notin \bigcap\{\text{Tautologies}(\mathcal{L}_n) \mid n \geq 2, n \in \mathbb{N}\}$$

"Fuzzy" logics

$W = [0, 1]$

Question: How to define conjunction?

Answer: Desired conditions

$f : [0, 1]^2 \rightarrow [0, 1]$ such that:

- f associative and commutative
- for all $0 \leq A \leq B \leq 1$ and all $0 \leq C \leq 1$ we have $f(A, C) \leq f(B, C)$
- for all $0 \leq C \leq 1$ we have $f(C, 1) = C$.

Definition A function with the properties above is called a t-norm.

Examples of t-norms

Gödel t-norm	$f_G(x, y) = \min(x, y)$
Łukasiewicz t-norm	$f_{Ł}(x, y) = \max(0, x + y - 1)$
Product t-norm	$f_P(x, y) = x \cdot y$

Left-continuous t-norm

Definition. A t-norm f is left-continuous if for every $x, y \in [0, 1]$ and every sequence $\{x_n\}_{n\in\mathbb{N}}$ with $0 \leq x_n \leq x$ and $lim_{n\to\infty} x_n = x$ we have $lim_{n\to\infty} f(x_n, y) = f(x, y)$.

Left-continuous t-norm

Definition. A t-norm f is left-continuous if for every $x, y \in [0,1]$ and every sequence $\{x_n\}_{n\in\mathbb{N}}$ with $0 \leq x_n \leq x$ and $lim_{n\to\infty} x_n = x$ we have $lim_{n\to\infty} f(x_n, y) = f(x, y)$.

The following t-norms are left continuous:

Gödel t-norm	$f_G(x,y) = \min(x,y)$
Łukasiewicz t-norm	$f_Ł(x,y) = \max(0, x+y-1)$
Product t-norm	$f_P(x,y) = x \cdot y$

Left continuous t-norms

With every left continuous t-norm f we can associate the following operations:

- $x \circ_f y = f(x, y)$
- $x \oplus_f y = 1 - f(1 - x, 1 - y)$
- $x \Rightarrow_f y = \max\{z \mid f(x, z) \leq y\}$
- $\neg_f x = x \Rightarrow_f 0$

Remark: Left continuity ensures that $\max\{z \mid f(x, z) \leq y\}$ exists.

Validity: $D = \{1\}$

Left continuous t-norms

With every left continuous t-norm f we can associate the following operations:

- $x \circ_f y = f(x, y)$
- $x \oplus_f y = 1 - f(1 - x, 1 - y)$
- $x \Rightarrow_f y = \max\{z \mid f(x, z) \leq y\}$
- $\neg_f x = x \Rightarrow_f 0$

Łukasiewicz t-norm

$$x \circ_{Ł} y = \max(0, x + y - 1)$$
$$x \oplus_{Ł} y = 1 - \max(0, 1 - x - y)$$
$$x \Rightarrow_f y = \min(1, 1 - x + y)$$
$$\neg x = \min(1, 1 - x) = 1 - x$$

Left continuous t-norms

With every left continuous t-norm f we can associate the following operations:

- $x \circ_f y = f(x, y)$
- $x \oplus_f y = 1 - f(1 - x, 1 - y)$
- $x \Rightarrow_f y = \max\{z \mid f(x, z) \leq y\}$
- $\neg_f x = x \Rightarrow_f 0$

Łukasiewicz t-norm

$$x \circ_{\text{Ł}} y = \max(0, x + y - 1) \qquad x \wedge_{\text{Ł}} y = x \circ_{\text{Ł}} (x \Rightarrow y)$$

$$x \oplus_{\text{Ł}} y = 1 - \max(0, 1 - x - y) \qquad x \vee_{\text{Ł}} y = \neg_{\text{Ł}}((\neg_{\text{Ł}} x) \wedge_{\text{Ł}} (\neg_{\text{Ł}} y))$$

$$x \Rightarrow_f y = \min(1, 1 - x + y)$$

$$\neg x = \min(1, 1 - x) = 1 - x$$

Left continuous t-norms

With every left continuous t-norm f we can associate the following operations:

- $x \circ_f y = f(x, y)$
- $x \oplus_f y = 1 - f(1 - x, 1 - y)$
- $x \Rightarrow_f y = \max\{z \mid f(x, z) \leq y\}$
- $\neg_f x = x \Rightarrow_f 0$

Gödel t-norm

$$x \circ_G y = \min(x, y)$$

$$x \oplus_G y = \max(x, y)$$

$$x \Rightarrow_G y = \max\{z \mid x \wedge z \leq y\} = \begin{cases} 1 & \text{if } x \leq y \\ y & \text{if } x > y \end{cases}$$

$$\neg_G x = \max\{z \mid x \wedge z = 0) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x > 0 \end{cases}$$

Checking validity of formulae in fuzzy logics

Given: F formula in a t-norm based fuzzy logic formed with the operations $\{\circ, \oplus, \neg, \Rightarrow\}$ (and also $\vee, \wedge$ if definable)

Task: Check whether F is valid (a tautology)
i.e. whether for all $\mathcal{A} : X \rightarrow [0, 1]$, $\mathcal{A}(F) = 1$

Idea:
Assume that there exists $\mathcal{A} : X \rightarrow [0, 1]$ such that $\mathcal{A}(F) \neq 1$.
Derive a contradiction.

Let $P_1, \dots, P_n$ be the propositional variables which occur in F.

Check whether $\exists x_1, \dots, x_n F(x_1, \dots, x_m) \neq 1$ is satisfiable in
$\mathcal{A} = ([0, 1], \{\circ_f, \oplus_f, \neg_f, \rightarrow_f, \leftrightarrow_f\})$.

Example 1: Łukasiewicz logic $\text{Ł} = \mathcal{L}_{\alpha_1}$

F $\mathcal{F}$-formula, where $\mathcal{F} = \{\vee, \wedge, \circ, \neg, \rightarrow, \leftrightarrow\}$.

Let $P_1, \dots, P_n$ be the propositional variables which occur in F.

Check whether $\exists x_1, \dots, x_n F(x_1, \dots, x_m) \neq 1$ is satisfiable in

$$[0,1]_{\text{Ł}} = ([0,1], \{\vee, \wedge, \circ, \neg, \rightarrow\})$$

where $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$ are the operations induced by the t-norm $f_{\text{Ł}}(x, y) = \max(0, x + y - 1)$, i.e.:

(Def$_{\circ_{\text{Ł}}}$)	$x{+}y{<}1 \rightarrow x{\circ}y{=}0$	$x{+}y{\geq}1 \rightarrow x{\circ}y{=}x{+}y{-}1$
(Def$_{\vee}$)	$x{\leq}y \rightarrow x{\vee}y{=}y$	$x{>}y \rightarrow x{\vee}y{=}x$
(Def$_{\wedge}$)	$x{\leq}y \rightarrow x{\wedge}y = x$	$x{>}y \rightarrow x{\wedge}y = y$
(Def$_{\Rightarrow_{\text{Ł}}}$)	$x{\leq}y \rightarrow x{\Rightarrow}y{=}1$	$x{>}y \rightarrow x{\Rightarrow}y = 1{-}x{+}y$
(Def$_{\neg_{\text{Ł}}}$)	$\neg x = 1 - x$	

Example 1: Łukasiewicz logic Ł $= \mathcal{L}_{\alpha_1}$

F $\mathcal{F}$-formula, where $\mathcal{F} = \{\vee, \wedge, \circ, \neg, \rightarrow, \leftrightarrow\}$.

Remark: The following are equivalent:

(1) $F(x_1, \dots, x_m) \neq 1$ is satisfiable in $[0,1]_{Ł} = ([0,1], \{\vee, \wedge, \circ, \neg, \rightarrow\})$, where $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$ are the operations induced by the t-norm $f_{Ł}$

(2) $\mathrm{Def}_{Ł} \wedge F(x_1, \dots, x_m) \neq 1$ satisfiable in $[0,1]$.

$(\mathrm{Def}_{\circ_{Ł}})$	$x+y<1 \rightarrow x\circ y=0$	$x+y\geq 1 \rightarrow x\circ y=x+y-1$
$(\mathrm{Def}_{\vee})$	$x\leq y \rightarrow x\vee y=y$	$x>y \rightarrow x\vee y=x$
$(\mathrm{Def}_{\wedge})$	$x\leq y \rightarrow x\wedge y = x$	$x>y \rightarrow x\wedge y = y$
$(\mathrm{Def}_{\Rightarrow_{Ł}})$	$x\leq y \rightarrow x\Rightarrow y=1$	$x>y \rightarrow x\Rightarrow y = 1-x+y$
$(\mathrm{Def}_{\neg_{Ł}})$	$\neg x = 1 - x$	

Example

To show: $((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y)$ is a tautology

New task: $\text{Def}_{Ł} \wedge \underbrace{((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y) \neq 1}_{G_1}$ unsatisfiable

where

	$(\text{Def}_\vee)$	$x \leq y \rightarrow x \vee y = y$	$x > y \rightarrow x \vee y = x$
	$(\text{Def}_\wedge)$	$x \leq y \rightarrow x \wedge y = x$	$x > y \rightarrow x \wedge y = y$
	$(\text{Def}_{\circ_Ł})$	$x + y < 1 \rightarrow x \circ y = 0$	$x + y \geq 1 \rightarrow x \circ y = x + y - 1$
	$(\text{Def}_{\Rightarrow_Ł})$	$x \leq y \rightarrow x \Rightarrow y = 1$	$x > y \rightarrow x \Rightarrow y = 1 - x + y$

Example

New task: $\text{Def}_{Ł} \wedge \underbrace{((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y) \neq 1}_{G_1}$ unsatisfiable

where

$$
\begin{array}{lll}
(\text{Def}_\vee) & x \leq y \rightarrow x \vee y = y & x > y \rightarrow x \vee y = x \\
(\text{Def}_\wedge) & x \leq y \rightarrow x \wedge y = x & x > y \rightarrow x \wedge y = y \\
(\text{Def}_{\circ_Ł}) & x + y < 1 \rightarrow x \circ y = 0 & x + y \geq 1 \rightarrow x \circ y = x + y - 1 \\
(\text{Def}_{\Rightarrow_Ł}) & x \leq y \rightarrow x \Rightarrow y = 1 & x > y \rightarrow x \Rightarrow y = 1 - x + y
\end{array}
$$

1. Rename subterms starting with Ł-operators and expand definitions:

$$
\begin{array}{l|lll}
p = x \Rightarrow 0 & s \neq 1 & x \leq 0 \rightarrow x \Rightarrow 0 = 1 & x > 0 \rightarrow x \Rightarrow 0 = 1 - x + 0 \\
q = p \Rightarrow 0 & & p \leq 0 \rightarrow p \Rightarrow 0 = 1 & p > 0 \rightarrow p \Rightarrow 0 = 1 - p + 0 \\
r = x \vee y & & q \leq r \rightarrow q \Rightarrow r = 1 & q > r \rightarrow q \Rightarrow r = 1 - q + r \\
s = q \Rightarrow r & & x \leq y \rightarrow x \vee y = y & x > y \rightarrow x \vee y = x
\end{array}
$$

Example

New task: $\mathrm{Def}_Ł \wedge \underbrace{((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y) \neq 1}_{G_1}$ unsatisfiable

where

$$
\begin{array}{lll}
(\mathrm{Def}_\vee) & x{\leq}y \rightarrow x{\vee}y{=}y & x{>}y \rightarrow x{\vee}y{=}x \\
(\mathrm{Def}_\wedge) & x{\leq}y \rightarrow x{\wedge}y = x & x{>}y \rightarrow x{\wedge}y = y \\
(\mathrm{Def}_{\circ_Ł}) & x{+}y{<}1 \rightarrow x{\circ}y{=}0 & x{+}y{\geq}1 \rightarrow x{\circ}y{=}x{+}y{-}1 \\
(\mathrm{Def}_{\Rightarrow_Ł}) & x{\leq}y \rightarrow x{\Rightarrow}y{=}1 & x{>}y \rightarrow x{\Rightarrow}y = 1{-}x{+}y
\end{array}
$$

2. Replace terms starting with Ł-operations; SAT checking in [0, 1]

$$
\begin{array}{l|llll}
p = x \Rightarrow 0 & s \neq 1 & x \leq 0 \rightarrow p = 1 & x > 0 \rightarrow p = 1 - x + 0 \\
q = p \Rightarrow 0 & & p \leq 0 \rightarrow q = 1 & p > 0 \rightarrow q = 1 - p + 0 \\
r = x \vee y & & q \leq r \rightarrow s = 1 & q > r \rightarrow s = 1 - q + r \\
s = q \Rightarrow r & & x \leq y \rightarrow r = y & x > y \rightarrow r = x
\end{array}
$$

Reduction to checking constraints over [0, 1]

Reduction to checking satisfiability in [0, 1] of constraints in linear arithmetic (implications of LA expressions).

NP complete [Sonntag'85]

Similar techniques can be used also for Gödel logics (with the Gödel t-norm).

This method was first described (in a slightly more general context) in:

Viorica Sofronie-Stokkermans and Carsten Ihlemann,
"Automated reasoning in some local extensions of ordered structures."
Proceedings of ISMVL'07, IEEE Press, paper 1, 2007.

and (with full proofs) in

Viorica Sofronie-Stokkermans and Carsten Ihlemann,
"Automated reasoning in some local extensions of ordered structures."
Journal of Multiple-Valued Logics and Soft Computing
(Special issue dedicated to ISMVL'07) 13 (4-6), 397-414, 2007.

Example 1: Gödel logic

F $\mathcal{F}$-formula, where $\mathcal{F} = \{\vee, \wedge, \neg, \rightarrow, \leftrightarrow\}$.

Let $P_1, \dots, P_n$ be the propositional variables which occur in F.

Check whether $\exists x_1, \dots, x_n F(x_1, \dots, x_m) \neq 1$ is satisfiable in

$$[0,1]_G = ([0,1], \{\vee, \wedge, \circ, \neg, \rightarrow\})$$

where $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$ are the operations induced by the t-norm $f_{\mathrm{G}}(x, y) = \min(x, y)$, i.e.:

$$\begin{array}{rll}
(\mathrm{Def}_\circ) = (\mathrm{Def}_\wedge) & x \leq y \rightarrow x \wedge y = x & x > y \rightarrow x \wedge y = y \\
(\mathrm{Def}_\vee) & x \leq y \rightarrow x \vee y = y & x > y \rightarrow x \vee y = x \\
(\mathrm{Def}_\Rightarrow) & x \leq y \rightarrow x \Rightarrow y = 1 & x > y \rightarrow x \Rightarrow y = y \\
(\mathrm{Def}_\neg) & x = 0 \rightarrow \neg x = 1 & x > 0 \rightarrow \neg x = 0
\end{array}$$

Example

Check whether $((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y)$ is a tautology in the Gödel logic.
New task: $\mathrm{Def}_G \wedge \underbrace{((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y) \neq 1}_{G_1}$ satisfiable?

where $(\mathrm{Def}_\circ) = (\mathrm{Def}_\wedge)$ $\quad x \leq y \rightarrow x \wedge y = x \qquad x > y \rightarrow x \wedge y = y$

$(\mathrm{Def}_\vee)$ $\quad x \leq y \rightarrow x \vee y = y \qquad x > y \rightarrow x \vee y = x$

$(\mathrm{Def}_\Rightarrow)$ $\quad x \leq y \rightarrow x \Rightarrow y = 1 \qquad x > y \rightarrow x \Rightarrow y = y$

$(\mathrm{Def}_\neg)$ $\quad x = 0 \rightarrow \neg x = 1 \qquad x > 0 \rightarrow \neg x = 0$

Example

New task: $\mathrm{Def}_G \wedge \underbrace{((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y) \neq 1}_{G_1}$ satisfiable?

$$
\begin{array}{lrll}
\textit{where} & (\mathrm{Def}_\circ) = (\mathrm{Def}_\wedge) & x{\leq}y \rightarrow x{\wedge}y = x & x{>}y \rightarrow x{\wedge}y = y \\
& (\mathrm{Def}_\vee) & x{\leq}y \rightarrow x{\vee}y{=}y & x{>}y \rightarrow x{\vee}y{=}x \\
& (\mathrm{Def}_\Rightarrow) & x{\leq}y \rightarrow x{\Rightarrow}y{=}1 & x{>}y \rightarrow x{\Rightarrow}y = y \\
& (\mathrm{Def}_\neg) & x = 0 \rightarrow \neg x = 1 & x > 0 \rightarrow \neg x = 0
\end{array}
$$

1. Rename subterms starting with Ł-operators and expand definitions:

$$
\begin{array}{l|lll}
p = x \Rightarrow 0 & s \neq 1 & x \leq 0 \rightarrow x \Rightarrow 0 = 1 & x > 0 \rightarrow x \Rightarrow 0 = 0 \\
q = p \Rightarrow 0 & & p \leq 0 \rightarrow p \Rightarrow 0 = 1 & p > 0 \rightarrow p \Rightarrow 0 = 0 \\
r = x \vee y & & q \leq r \rightarrow q \Rightarrow r = 1 & q > r \rightarrow q \Rightarrow r = r \\
s = q \Rightarrow r & & x \leq y \rightarrow x \vee y = y & x > y \rightarrow x \vee y = x
\end{array}
$$

Example

New task: $\mathrm{Def}_G \wedge \underbrace{((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y) \neq 1}_{G_1}$ satisfiable?

$$
\begin{array}{llll}
\textit{where} \quad (\mathrm{Def}_\circ) = (\mathrm{Def}_\wedge) & x{\leq}y \rightarrow x{\wedge}y = x & & x{>}y \rightarrow x{\wedge}y = y \\
(\mathrm{Def}_\vee) & x{\leq}y \rightarrow x{\vee}y{=}y & & x{>}y \rightarrow x{\vee}y{=}x \\
(\mathrm{Def}_\Rightarrow) & x{\leq}y \rightarrow x{\Rightarrow}y{=}1 & & x{>}y \rightarrow x{\Rightarrow}y = y \\
(\mathrm{Def}_\neg) & x = 0 \rightarrow \neg x = 1 & & x > 0 \rightarrow \neg x = 0
\end{array}
$$

2. Replace terms starting with Ł-operations; SAT checking in [0, 1]

$$
\begin{array}{l|llll}
p = x \Rightarrow 0 & s \neq 1 & x \leq 0 \rightarrow p = 1 & x > 0 \rightarrow p = 0 \\
q = p \Rightarrow 0 & & p \leq 0 \rightarrow q = 1 & p > 0 \rightarrow q = 0 \\
r = x \vee y & & q \leq r \rightarrow s = 1 & q > r \rightarrow s = r \\
s = q \Rightarrow r & & x \leq y \rightarrow r = y & x > y \rightarrow r = x
\end{array}
$$

Satisfiable (e.g. by $\beta(x){=}\beta(y){=}\frac{1}{2}$), so $((x \Rightarrow 0) \Rightarrow 0) \Rightarrow (x \vee y)$ not tautology in Gödel logic.

Product logic

Similar techniques can be used also for the product logic
(with the product t-norm)

$\mapsto$ non-linearity (hence higher complexity)

Many-Valued Logics

- Many-valued logics (finitely-valued)

  History and Motivation
  Syntax /Semantics

  Functional completeness

  Automated reasoning: Tableaux, Resolution

  Applications: logic; verification

- Infinitely-valued logics

  Examples: Łukasiewics logics $\mathcal{L}_{\aleph_0}, \mathcal{L}_{\aleph_1}$
  description of the tautologies

  Fuzzy logics:
  - t-norms, Łukasiewics, Gödel, Product t-norm
  - Łukasiewics logic, Gödel logic, Product logic
  - Automated methods for checking validity