Logica (I&E)

najaar 2018

`http://liacs.leidenuniv.nl/~vlietrvan1/logica/`

Rudy van Vliet
kamer 140 Snellius, tel. 071-527 2876
rvvliet(at)liacs(dot)nl

college 8, maandag 29 oktober 2018

1.5. Normal forms
1.6. SAT solvers

Vier achter en vier op het middenveld kan nooit functioneren. Je driehoeken vallen weg. Je moet altijd driehoeken hebben, want alleen dan heb je constant twee afspeelmogelijkheden.

A slide from lecture 7:

1.5.3. Horn clauses and satisfiability

Example.

$(p \wedge q \wedge s \to p) \wedge (q \wedge r \to p) \wedge (p \wedge s \to s)$

$(p_2 \wedge p_3 \wedge p_5 \to p_{13}) \wedge (\top \to p_5) \wedge (p_5 \wedge p_{11} \to \bot)$

A slide from lecture 7:

Deciding satisfiability for Horn formulas

function HORN(ϕ)
/* precondition: ϕ is a Horn formula */
/* postcondition: HORN(ϕ) decides the satisfiability for ϕ */
begin function
  mark all occurrences of $\top$ in ϕ
  while there is a conjunct $P_1 \wedge P_2 \wedge \cdots P_{k_i} \rightarrow P'$ of ϕ
    such that all P_j are marked but P' is not **do**
      mark P'
  end while

  if $\bot$ is marked
  then return 'unsatisfiable'
  else return 'satisfiable'
end function

A slide from lecture 7:

Exercise 1.5: 15.

Apply algorithm `HORN` to each of these Horn formulas:

(a)

$$(p \wedge q \wedge w \to \bot) \wedge (t \to \bot) \wedge (r \to p) \wedge (\top \to r) \wedge (\top \to q) \wedge (u \to s) \wedge (\top \to u)$$

Theorem 1.47. The algorithm `HORN` is correct for the satisfiability decision problem of Horn formulas,
and has no more than $n + 1$ cycles in its while-statement if n is the number of atoms in ϕ.
In particular `HORN` always terminates on correct input.

Proof

- termination

Theorem 1.47. The algorithm `HORN` is correct for the satisfiability decision problem of Horn formulas,
and has no more than $n + 1$ cycles in its while-statement if n is the number of atoms in ϕ.
In particular `HORN` always terminates on correct input.

Proof

- termination
- correct answer

> All marked P are true for all valuations in which ϕ evaluates to T.

holds after any number of executions of the body of the while statement.

All marked P are true for all valuations in which ϕ evaluates to T.

1.6. SAT solvers

All marked subformulas evaluate to their mark value for all valuations in which ϕ evaluates to T.

A linear solver

Translate formulas into equivalent formulas without $\vee$ and $\rightarrow$.

$$
\begin{aligned}
T(p) &= p \\
T(\neg\phi) &= \neg T(\phi) \\
T(\phi_1 \wedge \phi_2) &= T(\phi_1) \wedge T(\phi_2) \\
T(\phi_1 \vee \phi_2) &= \ldots \\
T(\phi_1 \rightarrow \phi_2) &= \ldots
\end{aligned}
$$

A linear solver

Translate formulas into equivalent formulas without $\vee$ and $\rightarrow$.

$$
\begin{aligned}
T(p) &= p \\
T(\neg\phi) &= \neg T(\phi) \\
T(\phi_1 \wedge \phi_2) &= T(\phi_1) \wedge T(\phi_2) \\
T(\phi_1 \vee \phi_2) &= \neg(\neg T(\phi_1) \wedge \neg T(\phi_2)) \\
T(\phi_1 \rightarrow \phi_2) &= \neg(T(\phi_1) \wedge \neg T(\phi_2))
\end{aligned}
$$

Example 1.48.

$$\phi = p \wedge \neg(q \vee \neg p)$$

$T(\phi)$. . .

parse tree. . .

DAG. . .

marking. . .

Rules for flow of constraints...

Post-processing of marking...

Example.

Sequent

$$p \wedge q \to r \vdash p \to q \to r$$

is valid, iff

$$\vdash (p \wedge q \to r) \to p \to q \to r$$

is valid, iff

$$\phi = \neg((p \wedge q \to r) \to p \to q \to r)$$

is not satisfiable.

$T(\phi)\ldots$
DAG. . .
marking. . .

Complexity. . .

But. . .

1.6.2. A cubic solver

Example.

Is

$$(p \vee q \vee r) \wedge (p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p) \wedge (\neg p \vee \neg q \vee \neg r)$$

satisfiable?

$$\phi = (p \vee (q \vee r)) \wedge ((p \vee \neg q) \wedge ((q \vee \neg r) \wedge ((r \vee \neg p) \wedge (\neg p \vee (\neg q \vee \neg r)))))$$

$T(\phi)\ldots$

$$\phi = (p \vee (q \vee r)) \wedge ((p \vee \neg q) \wedge ((q \vee \neg r) \wedge ((r \vee \neg p) \wedge (\neg p \vee (\neg q \vee \neg r)))))$$

$$\begin{aligned} T(\phi) \;&=\; \neg(\neg p \wedge \neg(\neg(\neg q \wedge \neg r))) \wedge (\neg(\neg p \wedge \neg\neg q) \wedge (\neg(\neg q \wedge \neg\neg r) \\ &\quad \wedge (\neg(\neg r \wedge \neg\neg p) \wedge \neg(\neg\neg p \wedge \neg(\neg(\neg\neg q \wedge \neg\neg r)))))) \end{aligned}$$

marking. . .

test an unmarked node n with T. . .

For some unmarked node n:
Test n with T
Test n with F

- If both runs find contradictory constraints, then...

- Else
  - nodes with same mark in both runs: ...
  - test next unmarked node

Until...

Complexity. . .

Optimizations:

- If one run for tested node finds contradictory constraints, . . .
- If either run finds consistent, complete marking, . . .