universität innsbruck

SS 2024 lecture 13

Logic

Diana Gründlinger Aart Middeldorp Fabian Mitterwallner

Alexander Montag Johannes Niederhauser Daniel Rainer

Outline

Definition

CTL* formulas consist of state formulas, which are evaluated in states:

$$\varphi \ ::= \ \bot \mid \top \mid p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid \mathsf{A}[\alpha] \mid \mathsf{E}[\alpha]$$

and path formulas, which are evaluated along paths:

$$\alpha \ ::= \ \varphi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha) \mid (\alpha \to \alpha) \mid (\mathsf{X}\,\alpha) \mid (\mathsf{F}\,\alpha) \mid (\mathsf{G}\,\alpha) \mid (\alpha\,\mathsf{U}\,\alpha)$$

Definition

satisfaction of CTL* state formula φ in state $s \in S$ and path formula α with respect to path π in model $\mathcal{M} = (S, \to, L)$ is defined by induction on φ

Theorem

satisfaction of CTL* formulas in finite models is decidable

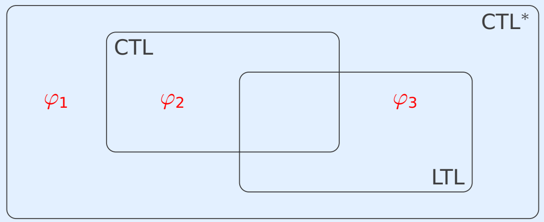
Definition

CTL* state (CTL, LTL) formulas φ and ψ are semantically equivalent if

$$\mathcal{M}, s \vDash \varphi \quad \Longleftrightarrow \quad \mathcal{M}, s \vDash \psi$$

for all models $\mathcal{M} = (S, \to, L)$ and states $s \in S$

Expressive Power



$\varphi_1 = \mathsf{E}[\mathsf{G}\,\mathsf{F}\,p]$

$\varphi_2 = \mathsf{AG}\,\mathsf{EF}\,p$

$\varphi_3 = \mathsf{A}[\mathsf{G}\,\mathsf{F}\,p \to \mathsf{F}\,q]$

Definition (Abstract DPLL)

- states $M \parallel F$ consist of list M of (annotated) non-complementary literals and CNF F
- transition rules
  - unit propagate $\qquad M \parallel F, C \vee \ell \implies M\,\ell \parallel F, C \vee \ell$

    if $M \vDash \neg C$ and ℓ is undefined in M
  - pure literal $\qquad M \parallel F \implies M\,\ell \parallel F$

    if ℓ occurs in F and ℓ^c does not occur in F and ℓ is undefined in M
  - decide $\qquad M \parallel F \implies M\,\overset{d}{\ell} \parallel F$

    if ℓ or ℓ^c occurs in F and ℓ is undefined in M
  - fail $\qquad M \parallel F, C \implies$ fail-state

    if $M \vDash \neg C$ and M contains no decision literals

Definition (Abstract DPLL, cont'd)

- backtrack $\qquad M\,\overset{d}{\ell}\,N \parallel F, C \implies M\,\ell^c \parallel F, C$

  if $M\,\overset{d}{\ell}\,N \vDash \neg C$ and N contains no decision literals
- backjump $\qquad M\,\overset{d}{\ell}\,N \parallel F, C \implies M\,\ell' \parallel F, C$

  if $M\,\overset{d}{\ell}\,N \vDash \neg C$ and there exists clause $C' \vee \ell'$ such that
  - $F, C \vDash C' \vee \ell'$ $\qquad$ backjump clause
  - $M \vDash \neg C'$
  - ℓ' is undefined in M
  - ℓ' or ℓ'^c occurs in F or in $M\,\overset{d}{\ell}\,N$

Definition

basic DPLL $\mathcal{B}$ consists of transition rules unit propagate, decide, fail, backjump

Lemma

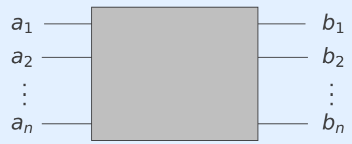
if $\parallel F \implies^*_{\mathcal{B}} M_0\,\overset{d}{\ell_1}\,M_1\,\overset{d}{\ell_2}\,M_2 \cdots \overset{d}{\ell_k}\,M_k \parallel F$ with no decision literals in $M_0, \dots, M_k$

then $F, \ell_1, \dots, \ell_i \vDash M_i$ for all $0 \leqslant i \leqslant k$

Theorem

- there are no infinite derivations $\parallel F \implies_{\mathcal{B}} S_1 \implies_{\mathcal{B}} S_2 \implies_{\mathcal{B}} \cdots$
- if $\parallel F \implies_{\mathcal{B}} S_1 \implies_{\mathcal{B}} \cdots \implies_{\mathcal{B}} S_n \not\Longrightarrow_{\mathcal{B}}$ then
  1. $S_n =$ fail-state if and only if F is unsatisfiable
  2. $S_n = M \parallel F'$ only if F is satisfiable and $M \vDash F$

Comparator Network



Definition

sorting network is comparator network that transforms any input sequence $a = (a_1, \dots, a_n)$ of natural numbers into sorted output sequence $b = (b_1, \dots, b_n)$:

b is permutation of a and $b_1 \leqslant \cdots \leqslant b_n$

Part I: Propositional Logic

algebraic normal forms, binary decision diagrams, conjunctive normal forms, DPLL, Horn formulas, natural deduction, Post's adequacy theorem, resolution, SAT, semantics, sorting networks, soundness and completeness, syntax, Tseitin's transformation

Part II: Predicate Logic

natural deduction, quantifier equivalences, resolution, semantics, Skolemization, syntax, undecidability, unification

Part III: Model Checking

adequacy, branching-time temporal logic, CTL*, fairness, linear-time temporal logic, model checking algorithms, symbolic model checking

Outline

Definition

function f is monotonic if $f(x) \leqslant f(y)$ whenever $x \leqslant y$

Lemma

if comparator network N transforms $(a_1,\dots,a_n)$ into $(b_1,\dots,b_n)$ and f is monotonic then N transforms $(f(a_1),\dots,f(a_n))$ into $(f(b_1),\dots,f(b_n))$

sketch

- induction on size
- if $x \leqslant y$ then $f(x) \leqslant f(y)$ and thus

$$\min(f(x),f(y)) = f(\min(x,y)) \qquad \max(f(x),f(y)) = f(\max(x,y))$$

Theorem (Zero–One Principle)

it suffices to test all 2^n sequences consisting of n zeros and ones

Proof (by contradiction)

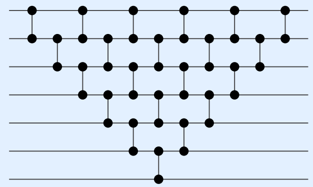
- suppose network sorts all binary sequences but not $a = (a_1,\dots,a_n) \in \mathbb{N}^n$
- $a_i < a_j$ but network places a_j before a_i in output sequence, for some i, j
- define function f as follows:

$$f(x) = \begin{cases} 0 & \text{if } x \leqslant a_i \\ 1 & \text{if } x > a_i \end{cases}$$

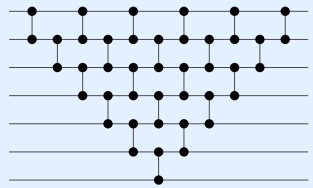
- f is monotonic
- consider binary sequence $f(a) = (f(a_1),\dots,f(a_n))$
- network places $f(a_j)$ before $f(a_i)$ when given $f(a)$ as input (by lemma)
- $f(a_j) = 1$ and $f(a_i) = 0$ ↯

Outline

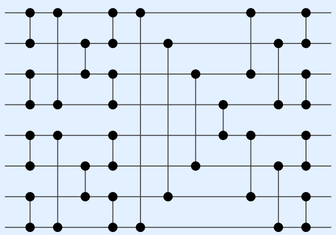
Insertion Sort



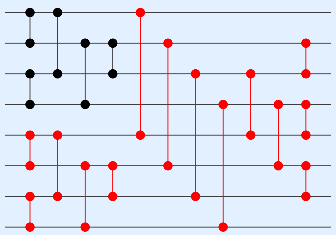
Bubble Sort



Bitonic Sort



Odd-Even Mergesort



Odd-Even Mergesort

to sort $(a_1, \dots, a_n)$ with $n = 2^m$ for some $m > 1$

① recursively sort $(a_1, \dots, a_{n/2})$ and $(a_{n/2+1}, \dots, a_n)$

② merge $(a_1, a_3, a_5, \dots, a_{n-1})$ and merge $(a_2, a_4, a_6, \dots, a_n)$

③ compare $(a_2, a_3), (a_4, a_5), \dots, (a_{n-2}, a_{n-1})$

Complexity

sorting n inputs requires

- insertion sort: depth $2n-3$ size $\frac{n(n-1)}{2}$
- bitonic sort: depth $\mathcal{O}(\log^2 n)$ size $\mathcal{O}(n \log^2 n)$
- odd-even mergesort: depth $\mathcal{O}(\log^2 n)$ size $\mathcal{O}(n \log^2 n)$

Outline

1. **Summary of Previous Lecture**
2. **Sorting Networks**

   Zero-One Principle Construction Optimality Verification
3. **Intermezzo**
4. **Conflict Graph**
5. **Further Reading**
6. **Course Recommendations**
7. **Exam**

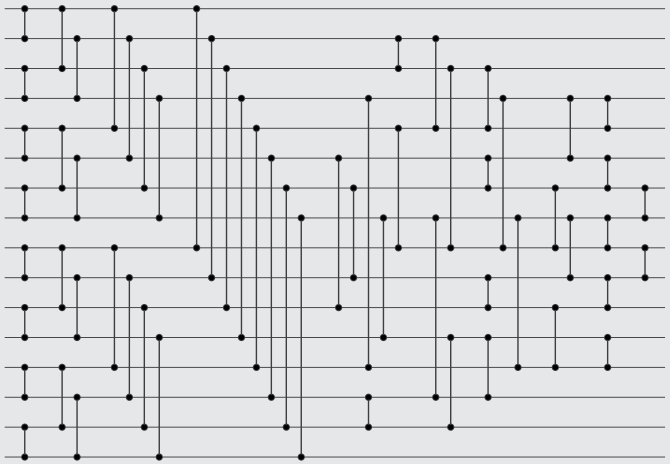
Known Bounds on Depth

# wires	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
upper bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9	10
lower bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9	10

Known Bounds on Size

# wires	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
upper bound	0	1	3	5	9	12	16	19	25	29	35	39	45	51	56	60	71
lower bound	0	1	3	5	9	12	16	19	25	29	33	37	41	45	49	53	58

Example (16 wires, size 60)



Outline

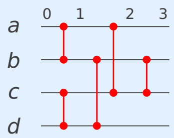
1. **Summary of Previous Lecture**
2. **Sorting Networks**

   Zero-One Principle Construction Optimality Verification
3. **Intermezzo**
4. **Conflict Graph**
5. **Further Reading**
6. **Course Recommendations**
7. **Exam**

Question

how to verify that comparator network is sorting network ?

SAT Encoding



variables x_i with $x \in \{a, b, c, d\}$ and $i \in \{0, 1, 2, 3\}$

constraints C

$$
\begin{array}{lll}
a_1 \leftrightarrow a_0 \wedge b_0 & a_2 \leftrightarrow a_1 \wedge c_1 & a_3 \leftrightarrow a_2 \\
b_1 \leftrightarrow a_0 \vee b_0 & b_2 \leftrightarrow b_1 \wedge d_1 & b_3 \leftrightarrow b_2 \wedge c_2 \\
c_1 \leftrightarrow c_0 \wedge d_0 & c_2 \leftrightarrow a_1 \vee c_1 & c_3 \leftrightarrow b_2 \vee c_2 \\
d_1 \leftrightarrow c_0 \vee d_0 & d_2 \leftrightarrow b_1 \vee d_1 & d_3 \leftrightarrow d_2
\end{array}
$$

sorting network $\iff$ $C \wedge ((a_3 \wedge \neg b_3) \vee (b_3 \wedge \neg c_3) \vee (c_3 \wedge \neg d_3))$ is unsatisfiable

Demo

SNV (sorting network visualizer)

by Rick Spiegl (2016)

Remark

sorting networks are used in pseudo-boolean constraint solving

$$x_1 + \cdots + x_n \geqslant 1 \qquad\qquad x_1 + \cdots + x_n \leqslant 1$$

Outline

particify with session ID **0992 9580**

Question

Which of the following statements are true ?

- **A** The function $f(x) = x^2$ is monotonic on the integers.
- **B** Given 2^n wires, there is a comparator network of depth n which computes minimum and maximum.
- **C** There is a sorting network for 20 wires with size 190 and depth 37.
- **D** The depth-optimal sorting network for a given size is unique.
- **E** If the sorting network constraint is satisfiable, an input which does not get sorted can be derived from a satisfying assignment.

Outline

see overlay version of slides for example of conflict graph

Remarks

- computed clauses are clauses that correspond to cut in conflict graph, separating conflict node from current decision literal and literals at earlier decision levels
- not all cuts are computed in this way
- clauses corresponding to UIPs are backjump clauses
- UIPs always exist (last decision literal)
- backjumping with respect to last UIP amounts to backtracking
- most SAT solvers use backjump clause corresponding to 1st UIP

Observation

adding backjump clauses to clause database (learning) helps to prune search space

- learn $\quad M \parallel F \implies M \parallel F, C$

  if $F \vDash C$ and each atom of C occurs in F or in M

Observation

restarts are useful to avoid wasting too much time in parts of search space without satisfying assignments

- restart $\quad M \parallel F \implies \parallel F$

Final Remarks

- restarts do not compromise completeness if number of steps between consecutive restarts strictly increases
- modern SAT solvers additionally incorporate
  - heuristics for selecting next decision literal
  - special data structures that allow for efficient unit propagation

Outline

Huth and Ryan

- Section 3.5

DPLL

- Section 2 of Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T)
  Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli
  Journal of the ACM 53(6), pp. 937–977, 2006
  doi: 10.1145/1217856.1217859
- Conflict-Driven Clause Learning SAT Solvers
  Joao Marques–Silva, Ines Lynce, and Sharad Mali
  Chapter 4 of Handbook of Satisfiability, IOS Press, 2008
  www.ics.uci.edu/~dechter/courses/ics-275a/winter-2016/readings/SATHandbook-CDCL.pdf

SAT

- The Silent (R)evolution of SAT
  Johannes K. Fichte, Daniel Le Berre, Markus Hecher, and Stefan Szeider
  Communications of the ACM 66(6), pp. 64–72, 2023
  doi: 10.1145/3560469

Sorting Networks

- Section 5.3.4 of The Art of Computer Programming
  Donald Knuth

Important Concepts

- bitonic sorting
- conflict graph
- cut
- learn
- odd-even mergesort
- restart
- unit propagation
- unique implication point (UIP)
- zero-one principle

homework for June 20

Grading — Proseminar

$$\text{score} = \min\left(\tfrac{50}{67}(E+P)+B, 100\right)$$

grade : $[0,50) \to \mathbf{5}$ $[50,63) \to \mathbf{4}$ $[63,75) \to \mathbf{3}$ $[75,88) \to \mathbf{2}$ $[88,100] \to \mathbf{1}$

Outline

Courses in 2024 / 2025

Program Verification	(WM 1)	VO3 + PS2	SS 2025
Term Rewriting	(PM 20)	VU3	SS 2025
Automata and Logic	(MSc WM 1)	VO2 + PS2	WS 2024
Constraint Solving	(MSc WM 2)	VO2 + PS2	SS 2025
Program and Resource Analysis	(MSc WM 8)	VU3	WS 2024
Tree Automata	(MSc WM 9)	VU3	WS 2024
Semantics of Programming Languages	(MSc WM 7)	VU3	SS 2025
Quantum Computation	(MSc WM 8)	VU3	SS 2025
Term Rewriting	(PM 20)	VU2	(Obergurgl)
Lambda Calculus and Type Theory	(PM 20)	VU2	(Obergurgl)

bachelor projects

Outline

First Exam on June 24

- 8:30 – 11:00 in HSB 3 and HSB 1 (email will follow)
- deregistration is possible until 23:59 on June 20
- closed book
- second exam on September 20, third exam on February 26, 2025

Preparation

- study previous exams
- review homework exercises and solutions
- study slides
- visit Tutorium — Wednesday, 16:15 – 17:00, SR 13
- visit consultation hours AM — Wednesday, 11:30 – 13:00, 3M07

evaluation SS 2024