Complexiteit 2023 — college 11

25 april 2023

Satisfiability

Vorige keer

- polynomiale reductie T van P naar Q: $P \leq_{\mathsf{P}} Q$
  - T beeldt elke invoer x van P af op een invoer $T(x)$ van Q
  - constructie van $T(x)$ uit x is polynomiaal (in $|x|$)
  - reductie-eigenschap: x is ja-instantie voor $P \iff T(x)$ is ja-instantie voor Q
  - hoop voorbeelden
- Q is NP-hard (NP-moeilijk) $\iff P \leq_{\mathsf{P}} Q$ voor alle $P \in \mathcal{NP}$
- Q is NP-volledig ($\mathcal{NPC}$) als Q NP-hard is en $Q \in \mathcal{NP}$
- NP-volledigheidsbewijs voor Q: bewijs dat $Q \in \mathcal{NP}$ en dat $P \leq_{\mathsf{P}} Q$ voor $P \in \mathcal{NPC}$
- SAT is het oer-$\mathcal{NPC}$-probleem

Vandaag

- Stelling van Cook–Levin: SAT is NP-moeilijk
- kort over de werking van SAT-solvers (geen tentamenstof)
- (Strong) Exponential Time Hypothesis (geen tentamenstof)

Nog even tussendoor

On the Complexity of Two Dots for Narrow Boards and Few Colors
`https://drops.dagstuhl.de/opus/volltexte/2018/8798/`
(FUN 2018)

Nog twee NP-volledigheidsbewijzen:

- een met maar drie kleuren (maar veel kolommen)
- een met maar twee kolommen (maar veel kleuren)

Two Dots is NP-volledig met drie kleuren

Exact Cover: gegeven een verzameling $U = \{u_1, \ldots, u_n\}$ en een verzameling van verzamelingen $F = \{S_1, \ldots, S_m\}$ (waar $|S_i| = 3$ voor alle i). Bestaat er een deelverzameling $G \subseteq F$ zodat elk element van U precies één keer voorkomt in de elementen van G?

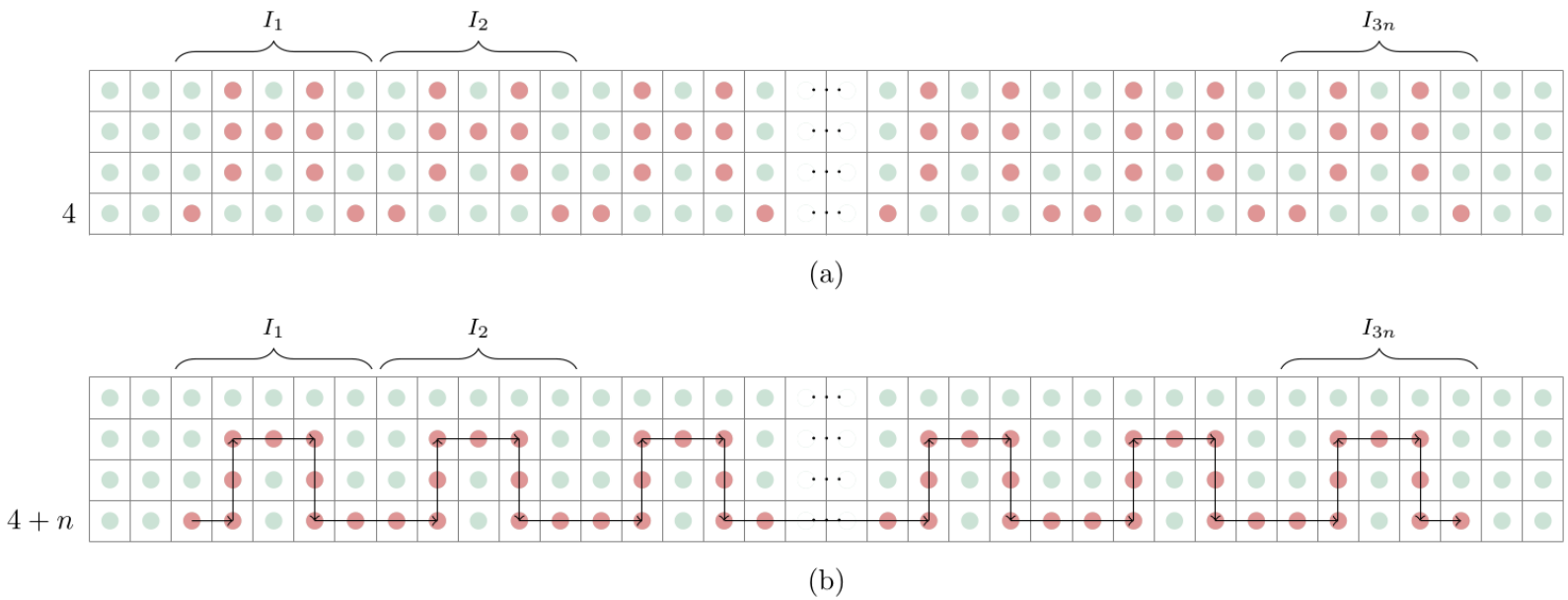


Figure 4 (a) Initial setup of the check-wire. (b) The check-wire once it gets aligned.

Two Dots is NP-volledig met drie kleuren

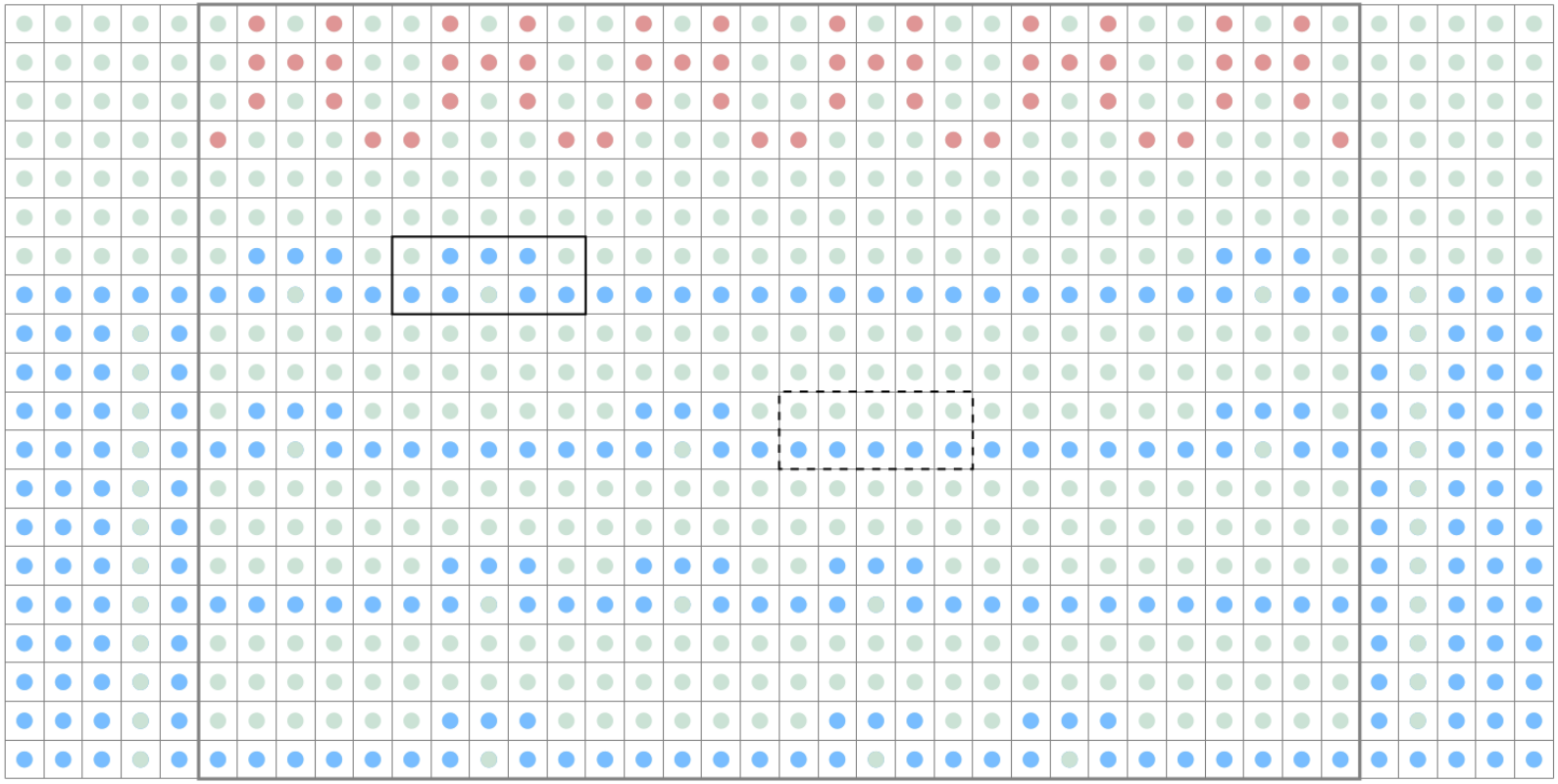


Figure 3 Overview of the reduction.

Doel: twee zetten voor een hoop blauw en al het rood.

Two Dots is NP-volledig met twee kolommen

Reductie vanaf **3SAT**.

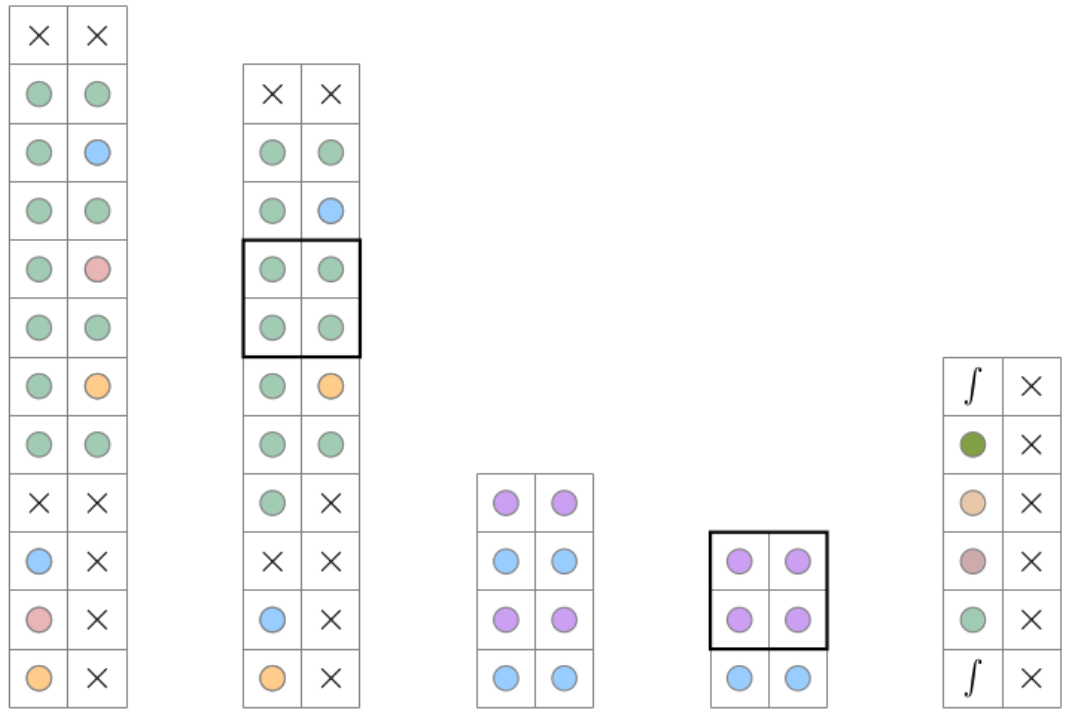


Figure 6 (a) Initial setup of the clause gadget. (b) The clause gadget in its aligned state. (c) Initial setup of the variable gadget. (d) The state of the variable gadget after one move on one of the literals. (e) The formula-check gadget.

Two Dots is NP-volledig met twee kolommen

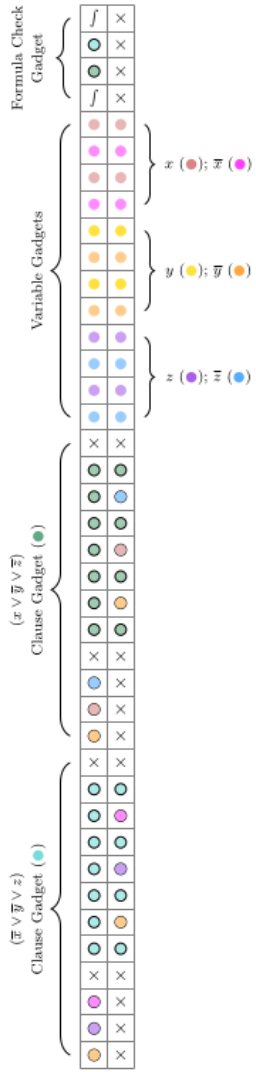


Figure 5 Overview of the reduction. The grid cells marked × are filled with distinct colors different from the ones used to represent variables and clauses. The goal of the game is to hit two dots of color ∫ and the number of moves are unbounded.

NP-volledigheid bewijzen

Stelling

Stel Q is een probleem waarvoor geldt dat $P \leq_{\mathsf{P}} Q$ voor een of andere $P \in \mathcal{NPC}$. Dan is Q NP-hard.
Als bovendien $Q \in \mathcal{NP}$, dan geldt dat $Q \in \mathcal{NPC}$.

Bewijs volgt uit de transitiviteit van $\leq_{\mathsf{P}}$ (twee slides terug) en de definitie van NP-hard (vorige slide).

M.a.w.: door een bekend NP-volledig probleem te reduceren tot Q reduceren we impliciet alle problemen uit $\mathcal{NP}$ tot Q. Dit geeft ons derhalve een methode om aan te tonen dat een probleem Q NP-volledig is.

Maar...

Deze reductiemethode is **inductief**:

$$\frac{P \in \mathcal{NPC} \quad P \leq_{\mathsf{P}} Q \quad Q \in \mathcal{NP}}{Q \in \mathcal{NPC}}$$

Wat ontbreekt hier nog?

Maar...

Deze reductiemethode is **inductief**:

$$\frac{P \in \mathcal{NPC} \quad P \leq_{\mathsf{P}} Q \quad Q \in \mathcal{NP}}{Q \in \mathcal{NPC}}$$

Wat ontbreekt hier nog?

Een **basisgeval**:

$$\frac{}{P_0 \in \mathcal{NPC}}$$

Stephen Cook

In 1971 bewees Stephen Cook op een directe manier (dus door een reductie te geven van alle problemen uit $\mathcal{NP}$) dat SAT NP-volledig is. (Details in volgend college.)

Stelling
Gegeven een *arbitrair* probleem $P \in \mathcal{NP}$. Dan is P reduceerbaar tot SAT: $P \leq_{\mathsf{P}}$ SAT.

Sindsdien is met behulp van de reductiemethode van veel bekende problemen aangetoond dat ze NP-volledig zijn. Bijvoorbeeld:

$$\begin{aligned} \text{SAT} \leq_{\mathsf{P}} \text{3SAT} \leq_{\mathsf{P}} \text{Kliek} \leq_{\mathsf{P}} \text{VC} \\ \text{3SAT} \leq_{\mathsf{P}} \text{HC2} \leq_{\mathsf{P}} \text{TSP} \\ \text{SAT} \leq_{\mathsf{P}} \text{3Kleur} \leq_{\mathsf{P}} \text{4Kleur} \end{aligned}$$

Stelling van Cook–Levin

Stephen Cook

Leonid Levin

Stelling (Cook 1971; Levin ±1973)
Gegeven een *willekeurig* probleem $P \in \mathcal{NP}$. Dan is P reduceerbaar tot SAT: $P \leq_{\mathsf{P}} \mathsf{SAT}$.

Nu: het bewijs van Cook. *The complexity of theorem proving procedures*, `https://doi.org/10.1145/800157.805047`.
Of: `https://www.cs.toronto.edu/~sacook/` (zoek zelf verder)

Even tussendoor: LaTeX

LaTeX stamt uit 1984, TeX uit 1978.

The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles

Even tussendoor: LaTeX

Definition: We will denote deg ({0}) by $\mathcal{L}_*$, where 0 denotes the zero function.

Thus $\mathcal{L}_*$ is the class of sets recognizable in polynomial time. $\mathcal{L}_*$ was discussed in [2], p. 5, and is the string analog of Cabham's class of functions [3].

We now define the following special sets of strings.

Even tussendoor: LaTeX

predicate calculus. Further, if M halts in s steps, then

$\phi(A(M)) \leq s^2$. Thus, if, contrary to (2), $T_Q(k) = 0(\sqrt{k}/\log^2 k)$, then a modification of M_Q could verify in only

$$0(\sqrt{s^2}/\log^2 s^2) = 0(s/\log^2 s)$$

steps that M halted in s steps

Bewijs van Cook

Schets van het bewijs

1. Omdat $P \in \mathcal{NP}$, is er een niet-deterministisch algoritme A (een niet-deterministische Turingmachine) voor P. Verder is A polynomiaal begrensd ($O(|x|^k)$ op invoer x).
2. Dit algoritme zal voor elke invoer x van P worden gemodelleerd als een logische formule $\phi = T(x)$ in CN: deze ϕ beschrijft a.h.w. de berekening van A, werkend op x.
3. De formule ϕ is weliswaar lang, maar kan worden geconstrueerd in hooguit $O(|x|^\ell)$ stappen.
4. Voor een ja-instantie x vergt de executie van A hooguit $N = N(x) \leq c \times |x|^k$ stappen.
5. Een waarmakende waardering voor ϕ correspondeert precies met een executie van A die een "ja" produceert. Dus er is een waardering die ϕ waarmaakt $\Leftrightarrow$ x is een ja-instantie voor P.

Niet-deterministische Turingmachine

Een NDTM-programma (algemene beschrijving*) bevat:

- Γ: een eindige verzameling tape-symbolen (waaronder invoeralfabet Σ en blanco). Voorbeeld: $\Gamma = \{0, 1, B\}$.
- Q: een eindige verzameling toestanden, waaronder een begintoestand q_0 en twee eindtoestanden q_Y en q_N. Voorbeeld: $Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$.
- $\delta \subseteq (Q \setminus \{q_Y, q_N\}) \times \Gamma \times Q \times \Gamma \times \{-1, 0, 1\}$ een transitierelatie die bepaalt wat er kan gebeuren als in een bepaalde toestand een bepaald karakter wordt gelezen.

In de begintoestand staat de invoerstring x op plek 1 tot en met $|x|$ en de rest van de tape is blanco. Het programma start in toestand q_0 met de lees- en schrijfkop op positie 1.

*equivalent met ons model, waarin het niet-deterministische gedeelte is samengebracht in Fase 1)

Bewijs van Cook

Boolese variabelen in ϕ:

Q_i^q : op tijdstip i is de machine in toestand $q \in Q$; $0 \leq i \leq N$

H_{ij} : op tijdstip i scant de machine cel j; $0 \leq i \leq N$, $-N \leq j \leq N$

S_{ij}^a : op tijdstip i bevat cel j symbool $a \in \Gamma$; $0 \leq i \leq N$, $-N \leq j \leq N$

De formule ϕ is een conjunctie van:

- $Q_0^{q_0} \wedge H_{01}$

  de machine start in toestand q_0, op positie 1 — 2 clausules

- $S_{0j}^{x_j}$, $1 \leq j \leq |x|$ $\quad$ S_{0j}^{B}, $-N \leq j \leq 0$ of $|x| < j \leq N$

  x op de posities 1 t/m $|x|$; rest bevat B — $2N + 1$ clausules

Bewijs van Cook

- $Q_N^{q_Y}$

  op tijdstip N stopt de berekening in de ja-toestand*

- $(\bigvee_{q \in Q} Q_i^q)$, $(\neg Q_i^p \vee \neg Q_i^q)$ $\quad 0 \leq i \leq N$, $p, q \in Q, p \neq q$

  altijd precies één toestand $\quad (N+1)\frac{|Q|(|Q|+1)}{2}$ clausules

- $(\bigvee_{a \in \Gamma} S_{ij}^a)$, $(\neg S_{ij}^a \vee \neg S_{ij}^b)$ $\quad 0 \leq i \leq N, -N \leq j \leq N$, $a, b \in \Gamma, a \neq b$

  cel precies één symbool $\quad (N+1)(2N+1)\frac{|\Gamma|(|\Gamma|+1)}{2}$ clausules

- $(\bigvee_{0 \leq j \leq N} H_{ij})$, $(\neg H_{ij} \vee \neg H_{ik})$ $\quad 0 \leq i \leq N$, $-N \leq j < k \leq N$

  scant precies één cel $\quad (N+1)(1+N(2N+1))$ clausules

*we mogen aannemen dat de machine in q_Y blijft als hij daar al eerder komt

Bewijs van Cook

- $Q_i^p \wedge H_{ij} \wedge S_{ij}^a \rightarrow \bigvee_{(p,a,q,b,d)\in\delta}(Q_{i+1}^q \wedge H_{i+1,j+d} \wedge S_{i+1,j}^b)$

  elke stap van de machine verloopt volgens de transitierelatie δ
  (nog wel "even" omschrijven naar CNF)

- $S_{ij}^a \wedge \neg H_{ij} \rightarrow S_{i+1,j}^a$

  een cel die op tijdstip i niet wordt gescand, bevat op tijdstip $i + 1$ hetzelfde symbool

Een waarmakende waardering komt zo precies overeen met een echte executie van de niet-deterministische Turingmachine die eindigt in "ja" na een polynomiaal (nl. N) aantal stappen.

“even” omschrijven en tellen

Voor de liefhebber.

$a \wedge b \to c \iff \neg a \vee \neg b \vee c$

$a \wedge b \wedge c \to \bigvee_{i \in I}(d_i \wedge e_i \wedge f_i) \iff (\neg a \vee \neg b \vee \neg c \vee \bigvee_{i \in I} z_i) \wedge \bigwedge_{i \in I}(\neg z_i \vee d_i) \wedge \bigwedge_{i \in I}(\neg z_i \vee e_i) \wedge \bigwedge_{i \in I}(\neg z_i \vee f_i)$ met verse variabelen z_i voor $i \in I$.

Als a, b en c alle drie waar zijn, moet er een z_i waar zijn. Vanwege de andere clausules moeten nu d_i, e_i en f_i waar zijn.

Aantal variabelen en clausules

Merk op dat $N \in O(|x|^k)$. Polynomiaal in N impliceert dus polynomiaal in $|x|$.

Verder zijn Q, Γ en δ deel van de invoer, dus polynomiaal in $|Q|, |\Gamma|en|\delta|$ is ook polynomiaal in $|x|$.

Verdere details: zie college.

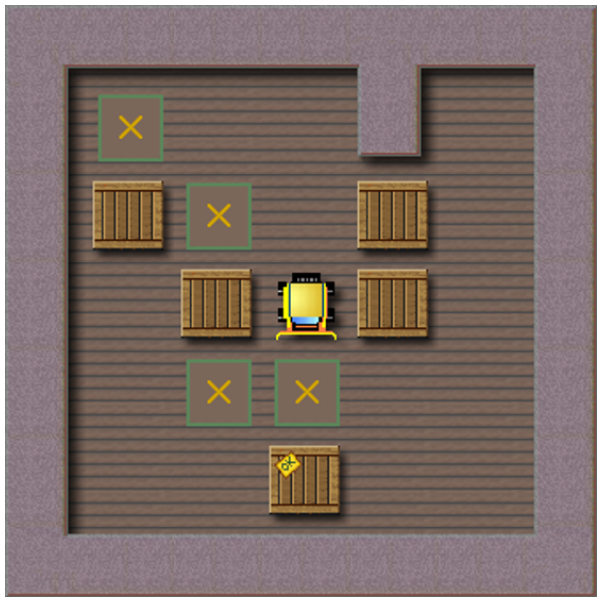
Reducties naar SAT: Sudoku (statisch)

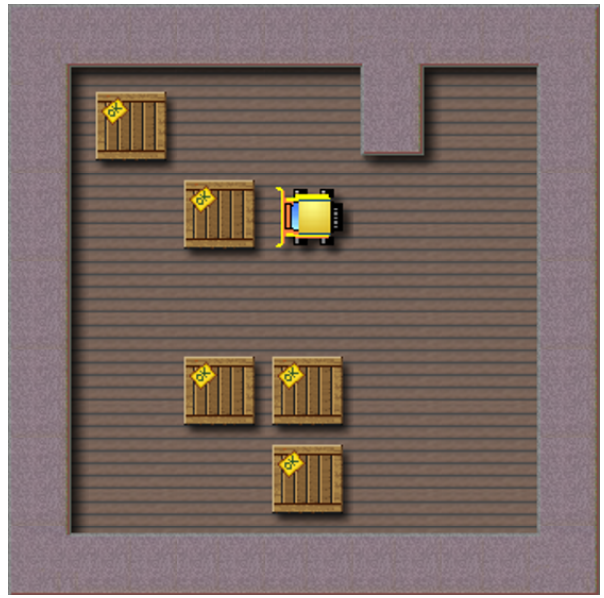
7	6		9					
			2	5			4	
	3		4		8	6		
4						5		8
	8		3	4	5		9	
1		9						4
		6	7		3		2	
	1			9	2			
					4		7	5

Variabelen: $x_{ij}^{k} \iff$ vakje (i, j) bevat cijfer k

- elk vakje bevat precies één cijfer
- rijen en kolommen mogen geen dubbelen bevatten
- de negen groepjes van 3×3 mogen geen dubbelen bevatten
- definieer een beginconfiguratie

Reducties naar SAT: Sokoban (dynamisch)





Idee (cf. reductie van Cook):

- definieer begintoestand en eindtoestand
- definieer geldige transitie tussen toestanden
- gebruik een of andere bovengrens voor het aantal stappen
- `https://cspsat.gitlab.io/copris-puzzles/sokoban/`

SAT-solvers

SAT-solvers zijn algoritmen of programma's die SAT-instanties oplossen, d.w.z., bepalen of ze ja- of nee-instanties zijn.

Een triviaal algoritme (exhaustive search) is te geven in $O(2^n \times |x|)$. Maar: kan het beter?

Handbook of Satisfiability (2009). Armin Biere, Marijn Heule, Hans van Maaren en Toby Walsh.
`https://books.google.nl/books?isbn=160750376X`

DPLL

Martin Davis, Hilary Putnam, George Logemann, Donald W. Loveland (1960/1961)

- Een van de meest bekende algoritmen, nog steeds de basis voor vele moderne
- Backtracking
- Kiest een literal, probeert een waardering, propageert de waarde door de formule en kijkt recursief of de vereenvoudige formule is waar te maken. Zo nee, probeer de andere waardering.
- *Unit propagation*: als een clausule nog maar één literal heeft, staat de waarde daarvan vast
- *Pure literal elimination*: literals die maar op één manier voorkomen krijgen de waarde die die waar maakt

Niet-chronologisch backtracken

- DPLL backtrackt *chronologisch*: bij het vinden van een tegenspraak, gaat het algoritme één stap terug in de zoekboom.
- Als die voorgaande stap echter niet relevant was voor de gevonden tegenspraak, is het onnodig om daarvan de andere waardering ook te proberen. Dat kan handiger.
- *Niet-chronologisch* backtracken is hierop een verbetering: bij het vinden van een tegenspraak, kijkt het algoritme welke vorige stappen deze tegenspraak hebben veroorzaakt. Dan wordt teruggesprongen naar de laatste stap uit deze verzameling.

Conflict-driven clause learning

- Conflict-driven clause learning (CDCL) is hierop nog een verbetering.
- Bij het vinden van een tegenspraak, kijkt het algoritme welke vorige stappen deze tegenspraak hebben veroorzaakt. Hieruit wordt een nieuwe clausule afgeleid en toegevoegd aan het probleem, om die fout voortaan direct te voorkomen.
- Vervolgens wordt doorgaans teruggesprongen naar de laatste stap waarin *unit propagation* iets doet met deze nieuwe clausule.
- Soms kunnen toegevoegde clausules op een later moment weer worden verwijderd.

SAT-conferenties en -wedstrijden

`http://satisfiability.org/SAT23/`

`https://satcompetition.github.io/2023/`

Ondergrens SAT?

Gigantische verbeteringen, zowel algoritmisch als implementatietechnisch, maar:

Wat is de ondergrens?

Even tussendoor: niet-polynomiaal en exponentieel

Polynomiaal: $O(n^k)$ ($k \geq 0$)

Exponentieel: $O(c^n)$ ($c > 1$)

Als $\mathcal{P} \neq \mathcal{NP}$: NP-volledige problemen niet polynomiaal oplosbaar.
Wil echter niet zeggen dat ze per se exponentieel zijn.

Quasi-polynomiaal: groter dan polynomiaal, maar nog niet exponentieel. $2^{O(\log^c n)}$ (constante $c > 0$)
Bijvoorbeeld: graafisomorfisme (Babai, 2017)

Subexponentieel: meerdere definities, maar bijv. $2^{O(\sqrt{n})}$.

Exponential Time Hypothesis

Russell Impagliazzo en Ramamohan Paturi (1999/2001):

Hypothese (Exponential Time Hypothesis (ETH))
Er bestaat een constante $c > 1$ zodat 3SAT $\in \Omega(c^n)$.

M.a.w.: 3SAT heeft in de worst case een exponentiële ondergrens.

Hypothese (Strong Exponential Time Hypothesis (SETH))
Voor elke constante $c < 2$ bestaat er een k zodat kSAT $\in \Omega(c^n)$.

M.a.w.: er bestaat (asymptotisch gezien) geen beter algoritme voor algemene SAT-instanties dan exhaustive search.

SETH $\rightarrow$ ETH $\rightarrow \mathcal{P} \neq \mathcal{NP}$, maar ook interessante gevolgen als ¬SETH of ¬ETH

Hypotheses niet algemeen aangenomen als waar; beide nog niet bewezen of ontkracht.

Vandaag gezien

- het basisgeval in $\mathcal{NPC}$-bewijzen: de stelling van Cook–Levin
  - reductie van arbitrair $\mathcal{NP}$-probleem naar SAT
  - encodeer elke instantie van het probleem in (groot maar) polynomiaal begrensde logische formule
- SAT-solvers: slimme technieken om dit probleem op te lossen
  - DPLL: backtracking met slimme extra's (unit propagation, pure literal elimination)
  - conflict-driven clause learning (CDCL): leer uit tegenspraken en voeg gaandeweg nieuwe clausules toe
- Exponential Time Hypothesis: hypothese dat 3SAT een exponentiële ondergrens heeft in de worst case
- Strong Exponential Time Hypothesis: hypothese dat SAT algemeen niet beter kan dan 2^n

(Werk)college

Volgende college: dinsdag 2 mei, 09u00–10u45, zaal 407-409 (Snellius)

Werkcollege: zodadelijk van 11u00 tot 12u45, computerzalen 302–304 en 303 (Snellius, onderaan de trap)
Opgaven uit het dictaat: 66, 67, 68

De komende weken

- 2 mei: herhaling
  - hoorcollege: korte samenvatting semester, uitgebreider waar gewenst
  - werkcollege: representatieve opgaven uit het hele semester
- 9 mei
  - hoorcollege: gastcollege Hendrik Jan Hoogeboom, verdere complexiteitsklassen en spellen
  - werkcollege: huiswerk 3 en desgewenst herhaling
- 16 mei: niets
- 23 mei: oefententamen (hoorcollege), geen werkcollege
- 5 juni: tentamen

Tentamenstof komt binnenkort duidelijker op de website.