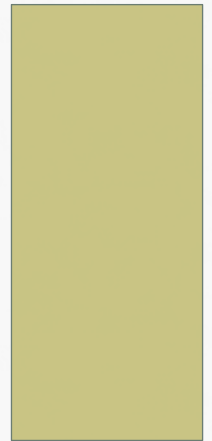


CERTIFYING ALGORITHMS

PRINZIP UND EINFACHE BEISPIELE

CHRISTIAN ACHENBACH



ÜBERSICHT

- Warnende Beispiele
- Geschichte und Entwicklung
- Bereits bekannte zertifizierende Algorithmen
- Definition und formales Framework
 - Strongly Certifying Algorithms
 - Beispiel: Bipartiter Graph
 - Beispiel: Planarer Graph in LEDA
 - Beispiel: Maximum-Matching
 - Certifying Algorithms
 - Beispiel: Binäre Suche
 - Weakly Certifying Algorithms
 - Beispiel: naiver randomisierter SAT-solver
- Vorteile

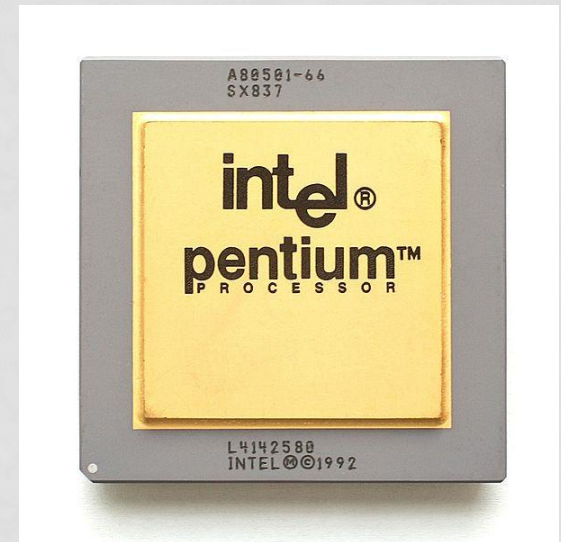
1996 - ARIANE 5



1994 - PENTIUM-FDIV-BUG

- Es handelte sich um einen Fehler in der FPU des Intel P5 CPUs.
- Verursacht wurde der Fehler durch 5 falsche Tabelleneinträge in einer Lookup-Tabelle.
- Beispiel der Auswirkung:

- $$\frac{824633702441}{824633702441} = 0,9999999996274709702$$



PLANARITY TEST IN LEDA 2.0

- LEDA 2.0 enthielt einen Fehler. Manche Graphen wurden falsch klassifiziert.
- In LEDA 2.1 wurde der Algorithmus korrigiert und durch einen zertifizierenden Algorithmus ersetzt.
- Zunächst hatte der Algorithmus eine quadratische Laufzeit. Dies wurde aber bald durch den Einsatz eines Algorithmus mit linearer Laufzeit verbessert.

GESCHICHTE

1967

- Robert Floyd
- Assertion

1969

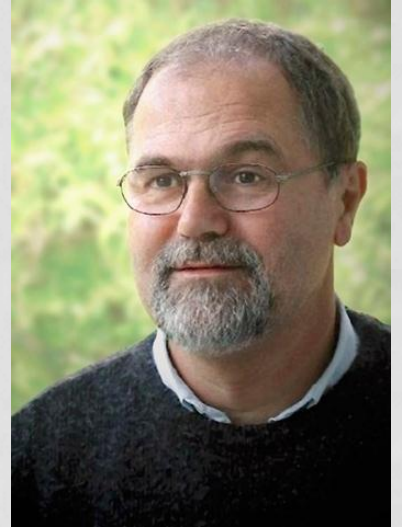
- C.A.R Hoare
- Hoare-Kalkül

'90er

- Clarke, Emerson, und Sifakis
- Model Checking

GESCHICHTE

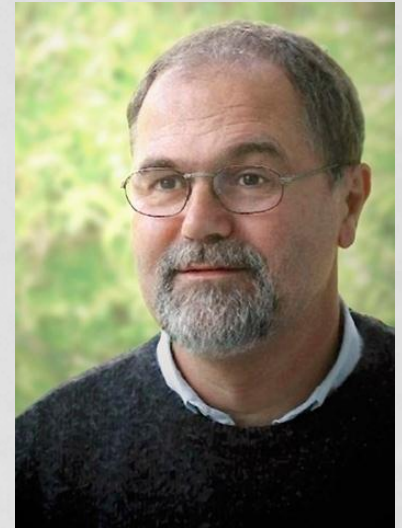
“programs must justify (prove) their answers in a way that is easily checked by their users.”



Prof. Dr.
Kurt Mehlhorn

GESCHICHTE

- 2006, erste Erwähnung von “Zertifizierende Algorithmen” durch Mehlhorn and Näher
- Über 100 bekannte “Zertifizierende Algorithmen”
- “LEDA contains checkers for all network and matching algorithms [...]for various connectivity problems on graphs, for planarity testing, for priority queues, and for the basic geometric algorithms”



Prof. Dr.
Kurt Mehlhorn

ERWEITERTER EUKLIDISCHER ALGORITHMUS

- Der allgemeine Euklidische Algorithmus ist nicht zertifizierend.
- Eine kleine Erweiterung macht ihn aber zu einem zertifizierenden Algorithmus:

$$\text{ggT}(a, b) = s * a + t * b$$

Beispiel:

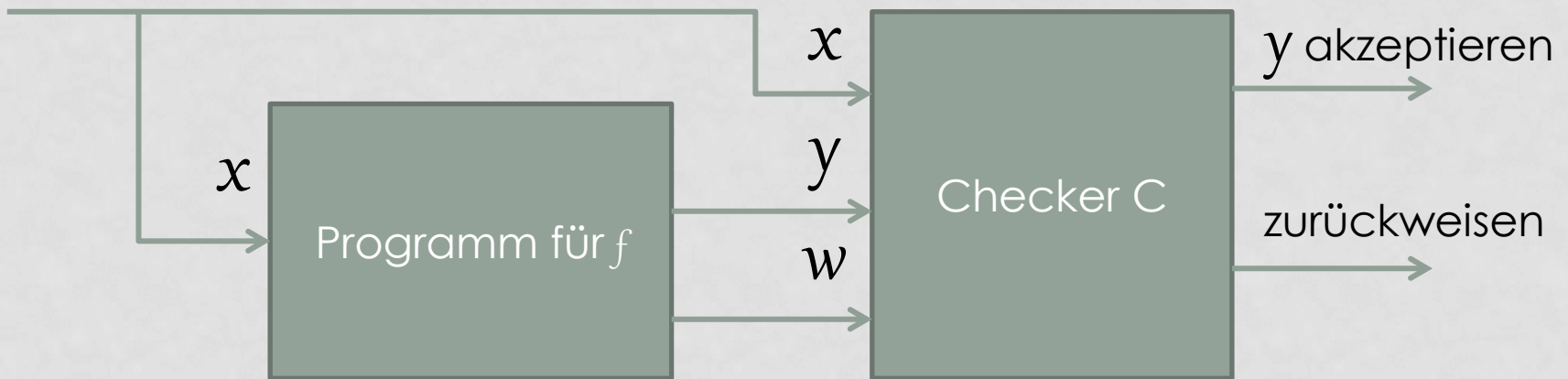
$$\text{ggT}(99, 78) = 3; s = -11; t = 14$$

NEUNERPROBE

Rechnung	Neunerprobe	
	Rest	Probe
573	6	6
*492	6	*6
281916	9	36=9
		9=9

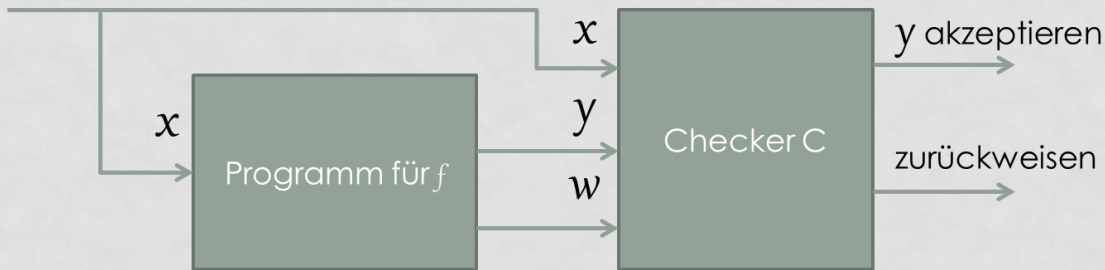


EINFÜHRUNG



Wichtig: Das Prinzip der Zertifizierenden Algorithmen basiert auf der Einfachheit und Korrektheit des Checkers.

DEFINITIONS AND FORMAL FRAMEWORK



Eingabe: $x \in X$

Ausgabe: $y \in Y$

Vorbedingung: $\varphi : X \rightarrow \{T, F\}$

Nachbedingung: $\psi : X \times Y \rightarrow \{T, F\}$

I/O-Spezifikation: (φ, ψ)

STRONGLY CERTIFYING ALGORITHMS

Erweiterte Ausgabe: $Y^\perp := Y \cup \{\perp\}$

\perp = Indikator für verletzte Vorbedingung

\mathcal{W} = Die Menge der Zeugen (witnesses)

(φ, ψ) hat die Struktur $\mathcal{W}: X \times Y^\perp \times \mathcal{W} \rightarrow \{T, F\}$

$$\forall x, y, w \quad \begin{array}{l} (y = \perp \wedge \mathcal{W}(x, y, w)) \Rightarrow \neg\varphi(x) \\ (y \in Y \wedge \mathcal{W}(x, y, w)) \Rightarrow \psi(x, y) \end{array}$$

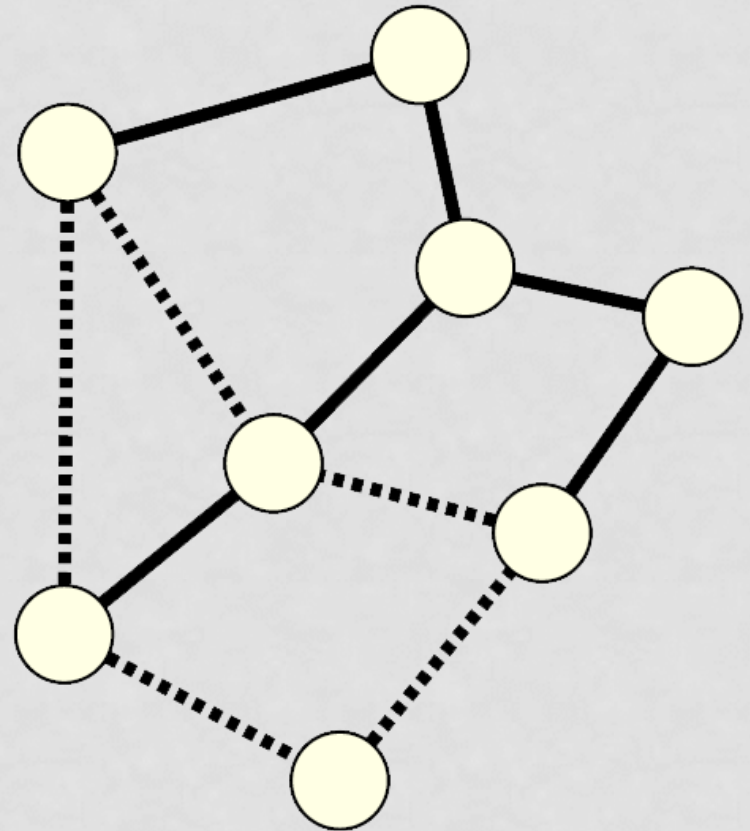
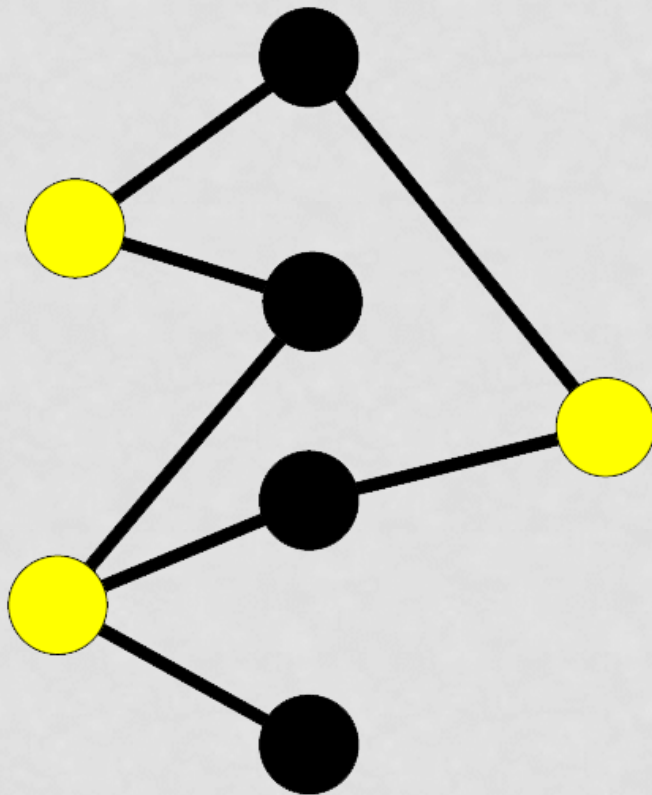
BEISPIEL: BIPARTITER GRAPH

- Definition: Ein einfacher Graph $G = (V, E)$ (V Menge der Knoten, E Menge der Kanten) heißt bipartit oder paar, falls sich seine Knoten in zwei disjunkte Teilmengen A und B aufteilen lassen, sodass zwischen den Knoten innerhalb beider Teilmengen keine Kanten verlaufen.

$Y = \{bipartite, nonbipartite\}$.

$is_bipartite : Menge\ aller\ endlichen\ Graphen \rightarrow \{0,1\}$

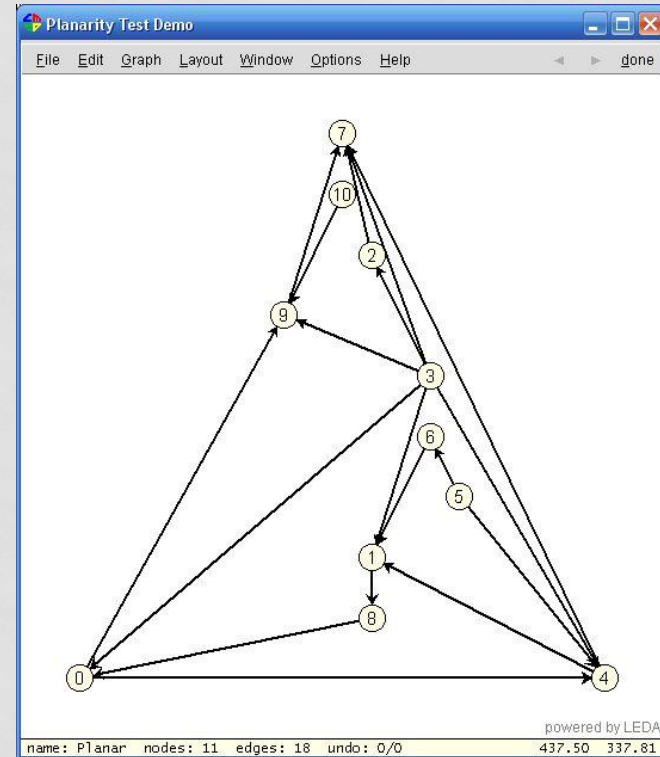
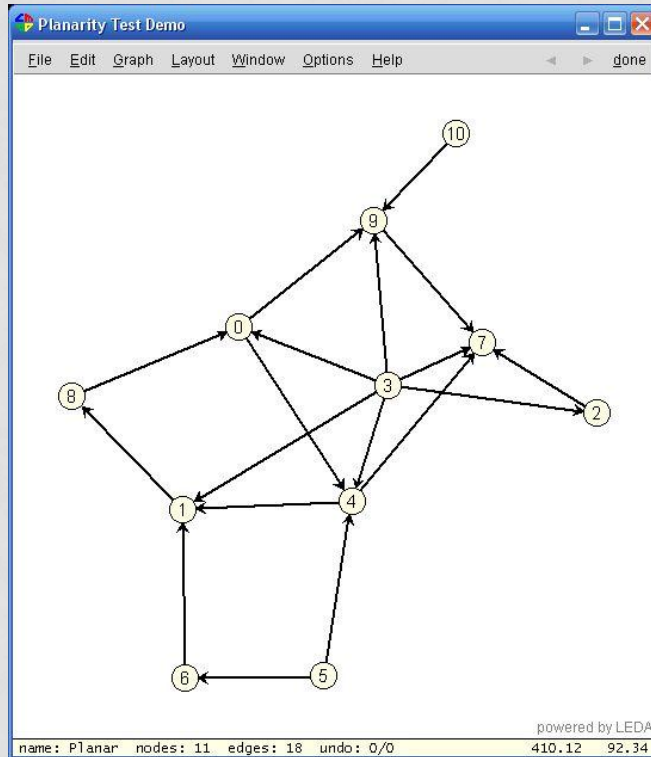
BEISPIEL: BIPARTITER GRAPH



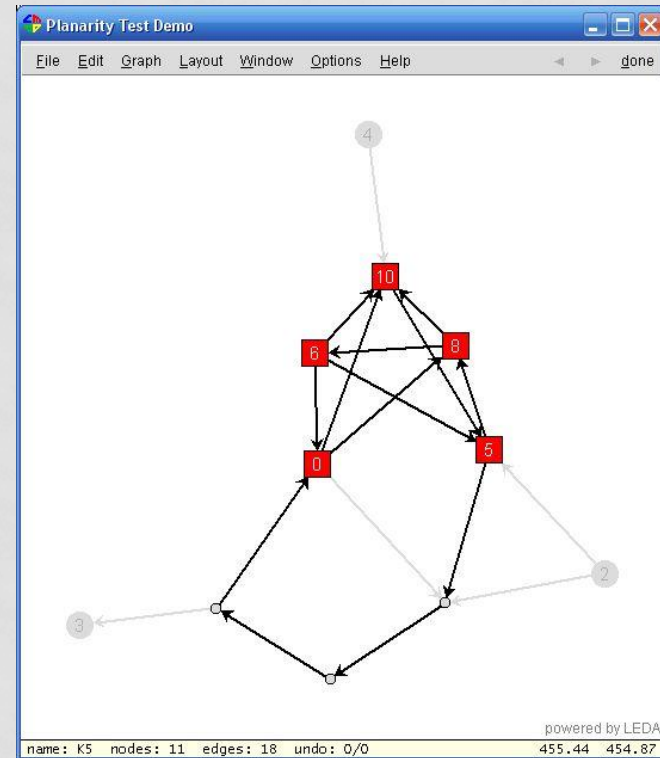
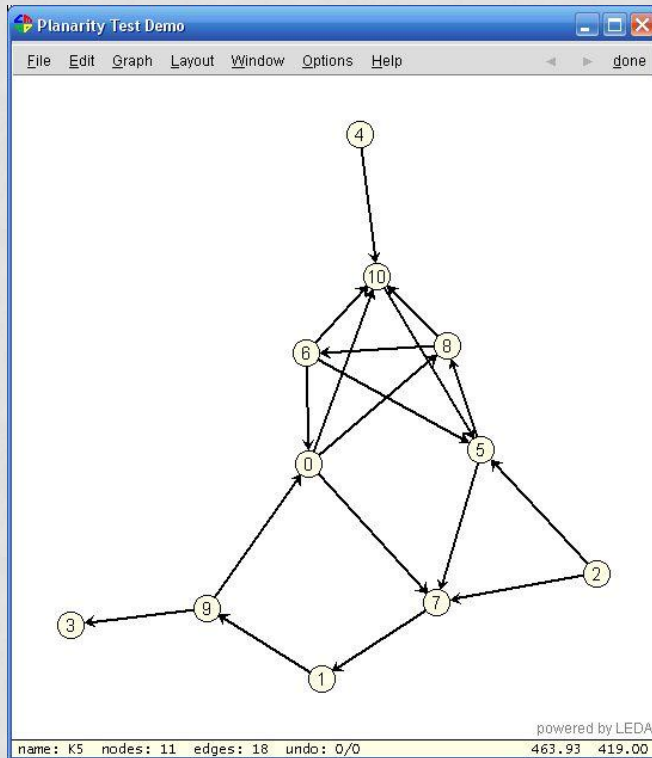
BEISPIEL: PLANARER GRAPH IN LEDA

- Definition: Ein Graph heisst planar, wenn man ihn so zeichnen (man sagt auch: in die Ebene einbetten) kann, dass sich keine Kanten kreuzen.
- Theorem: (Kuratowski) Ein Graph G ist genau dann planar, wenn G keinen Teilgraphen G' enthält, so dass K_5 oder $K_{3,3}$ ein Minor von G' ist.

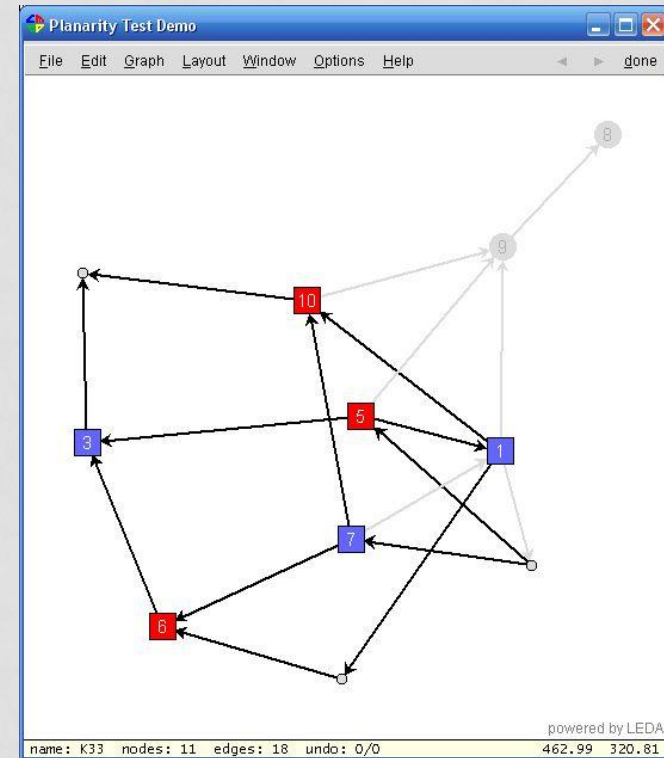
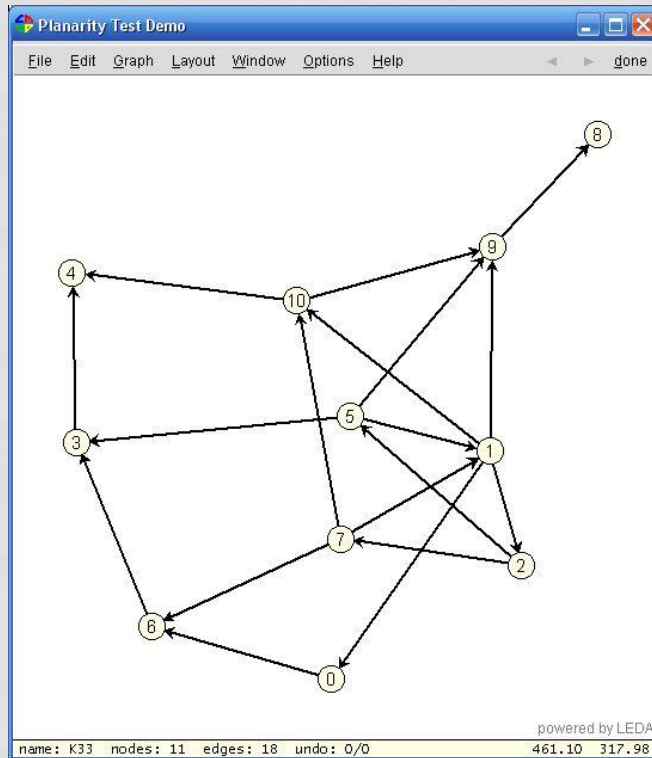
BEISPIEL: PLANARER GRAPH IN LEDA



BEISPIEL: PLANARER GRAPH IN LEDA



BEISPIEL: PLANARER GRAPH IN LEDA



(ORDINARY) CERTIFYING ALGORITHMS

$$\forall x, y, w \quad \begin{aligned} (y = \perp \wedge \mathcal{W}(x, y, w)) &\Rightarrow \neg\varphi(x) \\ (y \in Y \wedge \mathcal{W}(x, y, w)) &\Rightarrow \neg\varphi(x) \vee \psi(x, y) \end{aligned}$$

Gesucht wird „3“:

1	3	4	5	6	8	9	11	12
---	---	---	---	---	---	---	----	----

Gesucht wird „3“:

1	2	4	5	6	8	9	3	12
---	---	---	---	---	---	---	---	----

WEAKLY CERTIFYING ALGORITHMS

- Naiver randomisierter SAT-solver
- $\varphi(x)$ = (x ist eine erfüllbare boolesche Formel)
- Falls eine Belegung gefunden wird, dann Ausgabe: YES, ω als Beweis.
- Falls keine Belegung gefunden wird, dann läuft der Algorithmus eventuell unendlich lange.

VORTEILE

Belegte Korrektheit

- Falls $C(x, y, w)$ akzeptiert, dann beweist w entweder $\neg\varphi(x)$ oder $\psi(x, y)$
- Falls der $C(x, y, z)$ ablehnt, dann hat P y oder w falsch berechnet,

Tests über alle Eingaben

- Alle Eingaben x können getestet werden, nicht nur Eingaben mit bekannten Ausgaben.

VORTEILE

Fehler Identifizierung

- Fehlerquellen können identifiziert werden
- Ausbreitung des Fehlers kann verhindert werden.

Vertrauen ohne großen Aufwand

- Verständnis warum ein Zeuge die Korrektheit beweist.
- Verifizierung von C

VORTEILE

Entfernte Berechnung

- Berechnungen von einer nicht vertrauenswürdigen/störanfälligen Quellen können verifiziert werden.

Verifizierter Checker

- Formale Verifikation leicht zu erreichen.
- Auch in anderer Sprache implementierbar

RESÜMEE

“When you design your
next algorithm,
make it certifying”

Prof. Dr. Kurt Mehlhorn