Annäherung an das Nichtberechenbare
- Statische Programmanalyse -
Reinhard Wilhelm

Inhalt

- Eigenschaften von Programmen
- Approximation
- Effective Abstractions (P. Cousot)
- Voraussage des Cacheverhaltens

Interessante Fragen über Programme

Globale Eigenschaften:

- Terminierung für alle Eingaben
- Terminierung für spezielle Eingaben
- Realisierte Funktion

Interessante Fragen über Programme

Lokale Eigenschaften (an Programmpunkten)

- Erreichbarkeit
- Zustand (Werte der Programmvariablen)
- Invarianten: z.B. $x \leq y$, $x \neq 0$, $x \in [0,12]$
- Cacheinhalt bei Ausführung auf einem gegebenen Prozessor

Sicherheits- und Lebendigkeitseigenschaften

- Sicherheitseigenschaften: „something bad will not happen“
  Beispiel: Division durch 0, „index out of bounds“
- Lebendigkeitseigenschaften: „something good will happen“
  Beispiel: Programm reagiert auf Eingabe, Programmpunkt ist erreichbar

Entscheidungsverfahren

Eigenschaften:

- Vollständigkeit: Terminiert für alle Eingaben und gibt eine Antwort.
- Korrektheit: Alle Antworten stimmen.

Satz von Rice

Jede nichttriviale Eigenschaft der rekursiv aufzählbaren Sprachen ist unentscheidbar.

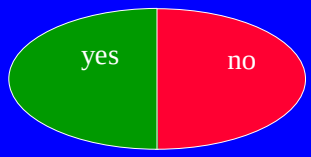
Folgerung 1: Alle nichttrivialen Eigenschaften von Programmen sind unentscheidbar.

Folgerung 2: Entscheidungsverfahren können nicht gleichzeitig vollständig und korrekt sein.

Kompromiss: Korrektheit, aber Unvollständigkeit „Erring on the safe side“

Approximation

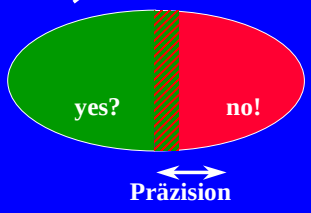
Wahre Antworten



Approximation

Sichere

~~Wahre~~ Antworten



Präzision

Beispiele für Approximationen

- Medizinischer Test auf eine Krankheit, i.a. weder vollständig noch korrekt
  - Falsch positiv: Diagnose „krank“, Patient gesund,
  - Falsch negativ: Diagnose „gesund“, Patient krank

Zusammenhang zwischen Testaufwand und Präzision

Beispiele für Approximationen II

- Neunerprobe: Die Richtigkeit von Berechnungen mit Addition, Subtraktion und Multiplikation wird auf den Quersummen überprüft.

```
  345   3
+ 246   3
-----------
  581   5 | 6
```

- Vorsicht! Auf welcher Seite sicher?

Beispiele für Approximationen III

Bestimmung des Vorzeichens von Ausdrücken:
(Abstrakte) Addition $+^{\#}$

$+^{\#}$	0	P	N	?
0	0	P	N	?
P	P	P	?	?
N	N	?	N	?
?	?	?	?	?

Beispiele für Approximationen

Abstrakte Multiplikation $*^{\#}$

$*^{\#}$	0	P	N	?
0	0	0	0	0
P	0	P	N	?
N	0	N	P	?
?	0	?	?	?

Formaler Hintergrund

Abstraktion α : konkrete Werte -> abstrakte Werte

Abstrakte Werte sind Beschreibungen (von Mengen) konkreter Werte

Beispiel: α_{sign} : Int -> {0, P, N, ?}

$\alpha_{sign}(7) = P$, $\alpha_{sign}(-7) = N$, $\alpha_{sign}(0) = 0$

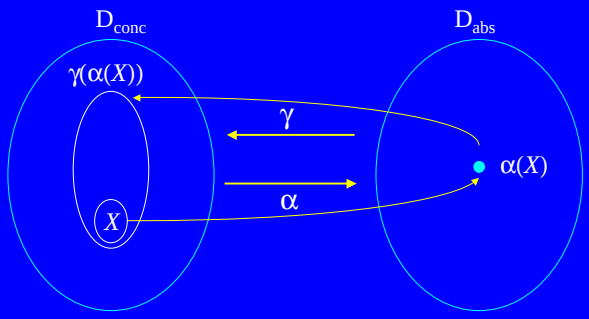
Konkretisierung γ : abstr. Werte -> konkr. Werte

Beispiel: γ_{sign} : {0, P, N, ?} -> 2^{Int}

$\gamma_{sign}(P) = \{1,2,3,...\}$, $\gamma_{sign}(N) = \{-1,-2,-3,...\}$,

$\gamma_{sign}(0) = \{0\}$, $\gamma_{sign}(?) =$?

Konkreter und Abstrakter Bereich



Informationsgehalt, Präzision

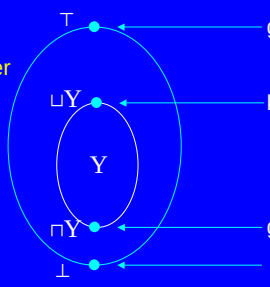
Abstrakte Elemente *a*, *b*

$\gamma(a) \subseteq \gamma(b) \Rightarrow$ *a* enthält bessere Information als *b*

Abstrakte Elemente in Verband anordnen.

Verbände

(L, $\sqsubseteq$)
Vollständiger Verband



größtes Element

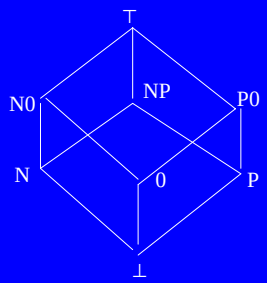
kleinste obere Schranke

größte untere Schranke

kleinstes Element

Partielle Ordnung, $\sqsubseteq$, gibt Präzision an,
"niedriger ist präziser"

(Feinerer) Verband für Vorzeichenanalyse



$\gamma(N) = \{-1,-2,...\} \subset \{0,-1,-2,...\} = \gamma(N0)$.
N ist also präziser als N0

Semantik, konkret und abstrakt

konkret

(X,-1)

X := X + 1

(X,0)

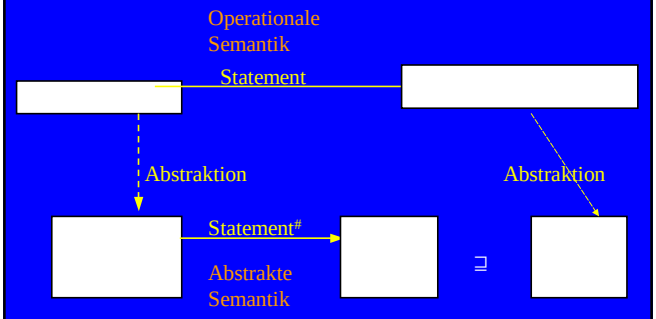
abstrakt

(X,N)

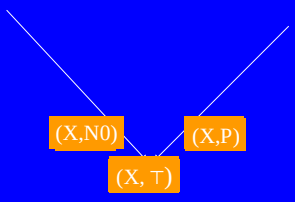
(X := X + 1)#

(X,?)

Lokale Korrektheit abstrakter Interpretation

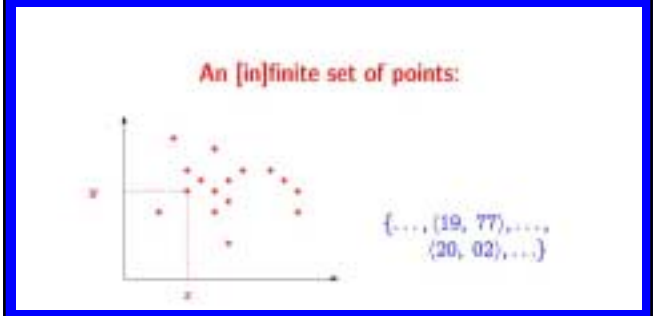


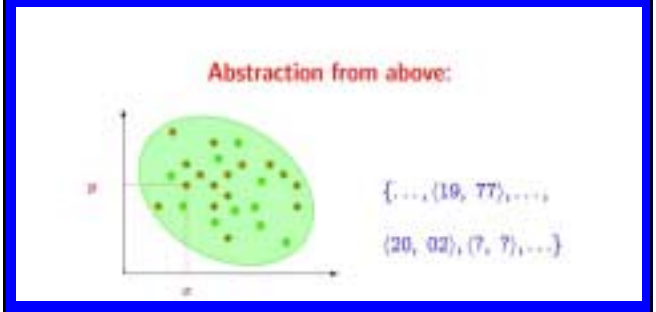
Kontrollzusammenfluss

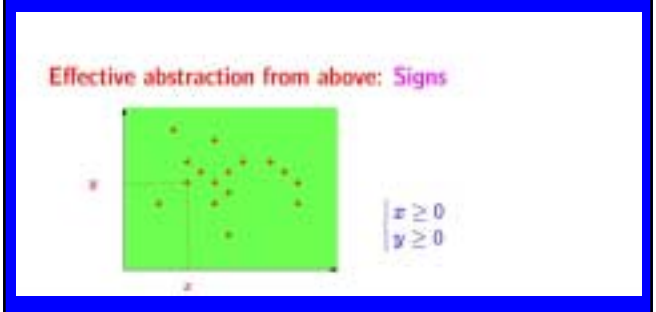


Operation auf den eingehenden abstrakten Werten:
Kleinste obere Schranke, ⊔.

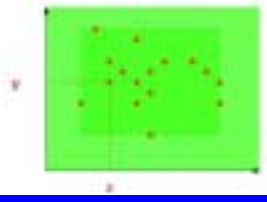
Effective Abstractions (P. Cousot)





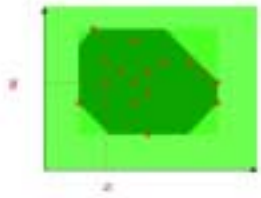


Effective abstraction from above: Intervals



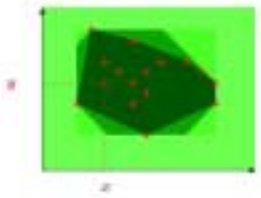
$$\begin{array}{l} x \in [19,\ 77] \\ y \in [20,\ 02] \end{array}$$

Effective abstraction from above: Octagons



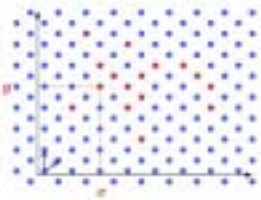
$$\begin{array}{l} 1 \leq x \leq 9 \\ x + y \leq 77 \\ 1 \leq y \leq 9 \\ x - y \leq 99 \end{array}$$

Effective abstraction from above: Polyhedra



$$\begin{array}{l} 19x + 77y \leq 2002 \\ 20x + 02y \geq 0 \end{array}$$

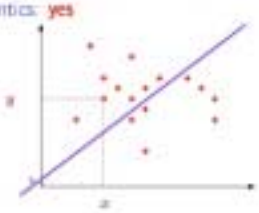
Effective abstraction from above: Linear congruences



$$\begin{array}{l} 1x + 9y = 7 \bmod 8 \\ 2x - 1y = 9 \bmod 9 \end{array}$$

Conservative Approximation

- Is the operation `1/(x+1-y)` well defined at run-time?
- Concrete semantics: **yes**

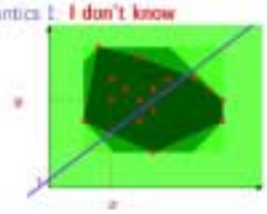


Conservative Approximation

- Is the operation `1/(x+1-y)` well defined at run-time?
- Testing: **You never know!**

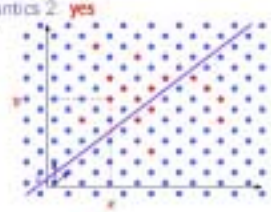
Conservative Approximation

- Is the operation 1/(x+1-y) well defined at run-time?
- Abstract semantics 1: I don't know



Conservative Approximation

- Is the operation 1/(x+1-y) well defined at run-time?
- Abstract semantics 2: yes



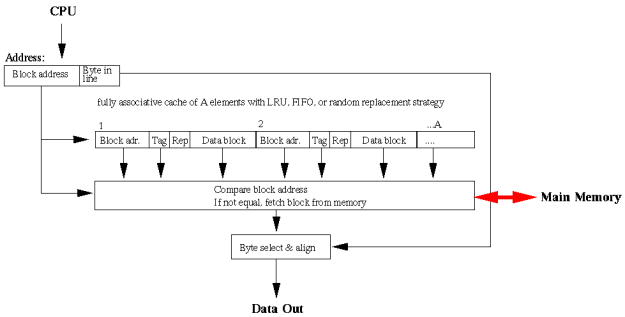
Hard Real-Time Systems

- Controllers in planes, cars, plants, … are expected to finish their tasks reliably within time bounds.
- Task scheduling must be performed
- Hence, it is essential that the Worst-Case Execution Time (WCET) of all program tasks is known.

Cache Memories

- Improve access times of fast microprocessors to slow memories
- Store recently referenced memory blocks (principle hope)
- Hit time ~ 1cycle; Miss penalty ~ 1-100 cycles
- 3 key parameter:
  Size, line length, level of associativity

Fully Associative Cache



Real-Time Systems

Oxford Dictionary of Computing:

Any system in which the time at which output is produced is significant.

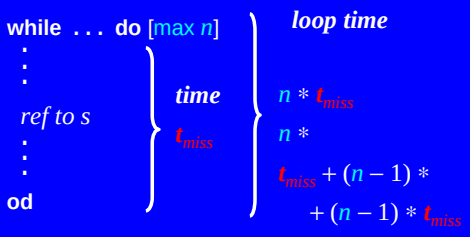
This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness.

Cache Memories and Real-Time Systems

- On hardware with caches: worst case assumption every access is a cache miss
- Worst case timings are far away from realistic timings. This leads to a waste of hardware resources.

WCET Computation

Information about cache contents sharpens timings.



Result of the Cache Analyses

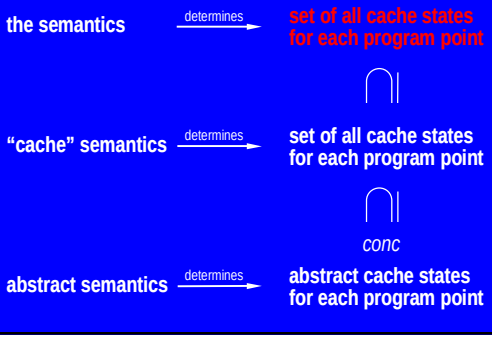
Category	Abb.	Meaning
always hit		The memory reference will always result in a cache hit.
always miss		The memory reference will always result in a cache miss.
not classified		The memory reference could neither be classified as nor .

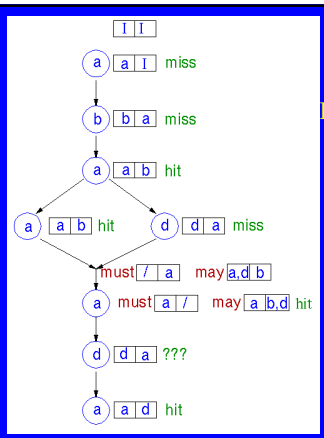
Cache Analysis

- Must Analysis:
  For each program point and calling context, find out which blocks are in the cache.
- May Analysis:
  For each program point and calling context, find out which blocks may be in the cache.

Cache Analysis II

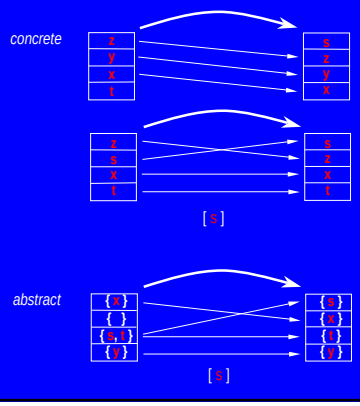
Approximation of the Collecting Semantics



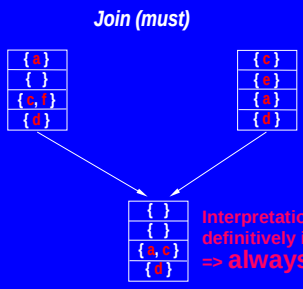


Example:
Fully Associative Cache (2 Elements)

Cache Analysis: Semantics



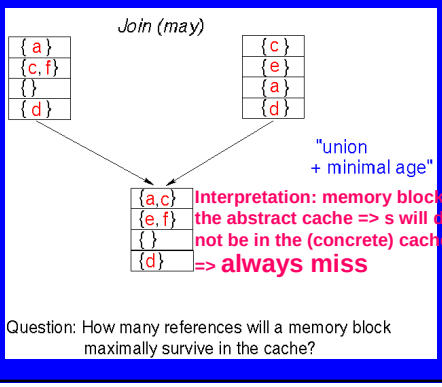
Cache Analysis: Abstract Semantics: Join (must)



Interpretation: memory block a is definitively in the (concrete) cache => always hit

Question: How many references will a memory block surely survive in the cache?

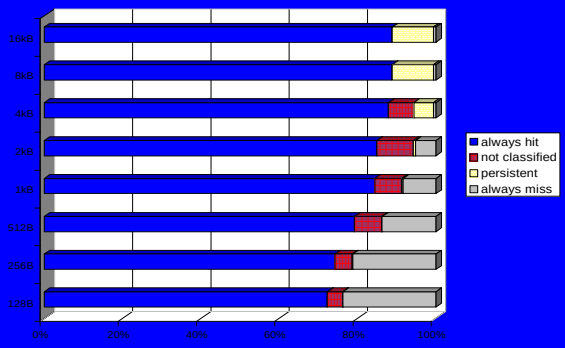
Cache Analysis: Abstract Semantics: Join (may)



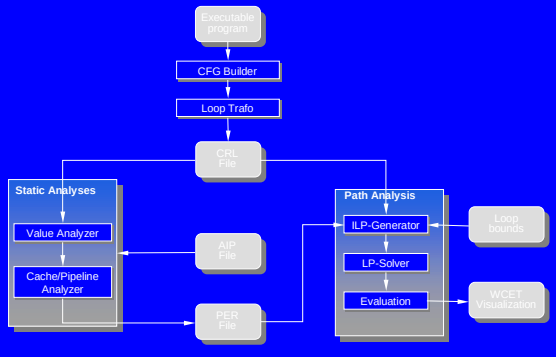
Interpretation: memory block s not in the abstract cache => s will definitively not be in the (concrete) cache => always miss

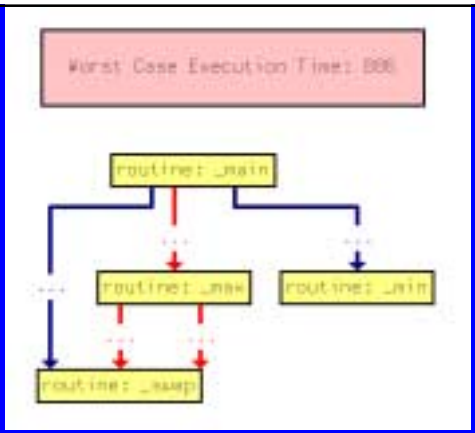
Question: How many references will a memory block maximally survive in the cache?

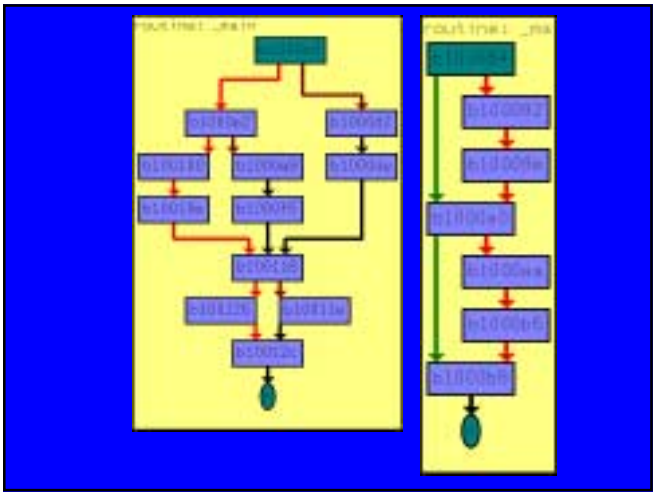
Experimental Results: djpeg on SPARC ISA 4-way set associative I-Cache 16 Bytes lines

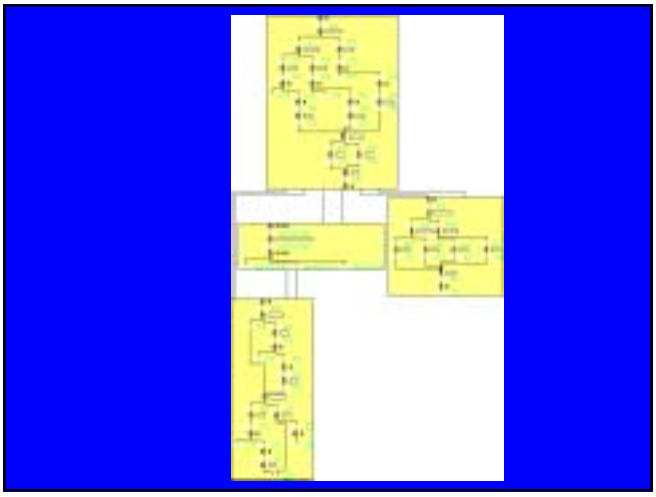


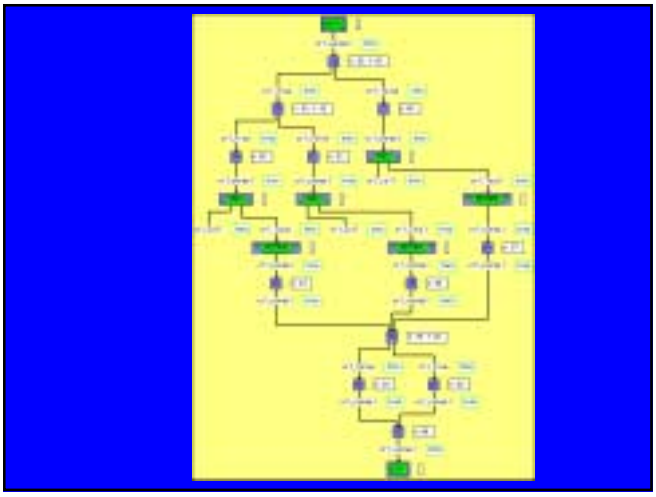
Overall Structure of WCET Tool

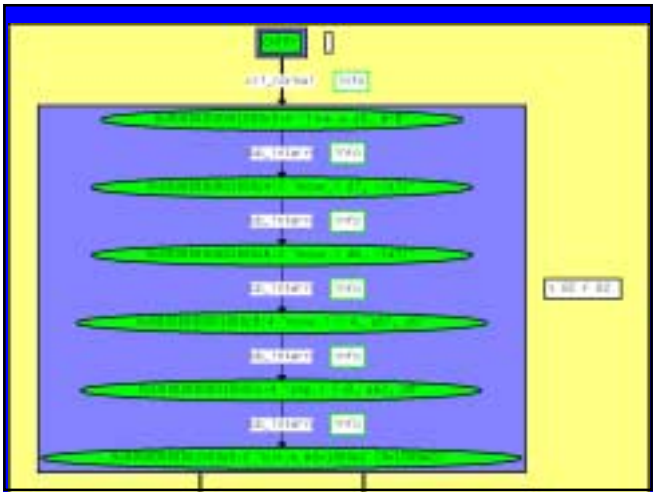


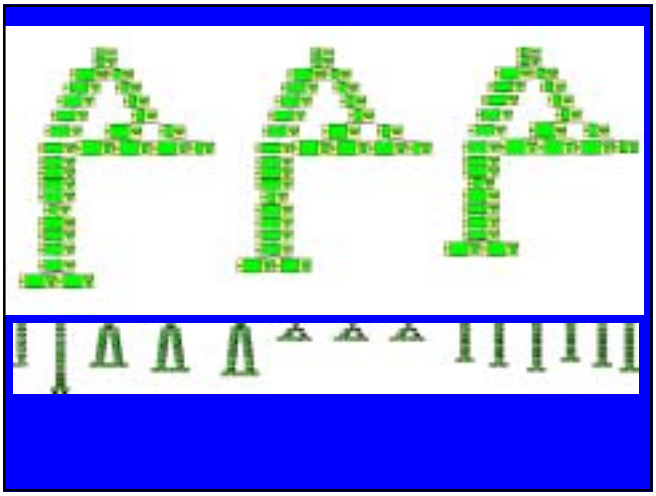


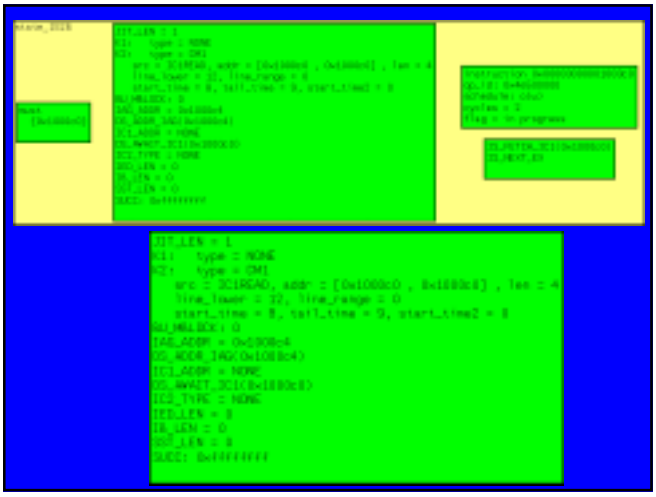












Zusamenfassung

- Abstraktion, wichtiges Prinzip in der Informatik
- Realistische Anwendung Cache-Analyse
- WCET-Tool hält Einzug in die Industrie