Karlsruhe Institute of Technology

Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

SATABS: Predicate Abstraction

Dr. Stephan Falke
Dr. Carsten Sinz

Institut für Theoretische Informatik

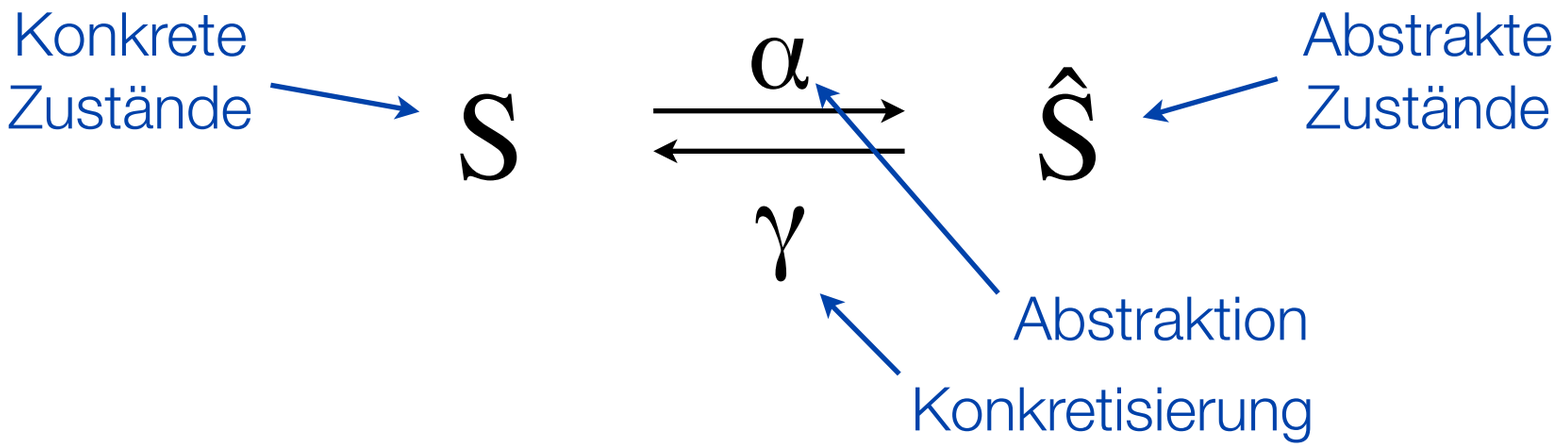
01.07.2013

Prädikatabstraktion (*Predicate Abstraction*)

- **Grundidee:**
  - Ausführung eines Programms auf „abstrakten Werten“
    - **Beispiel:** anstelle aller Werte einer ganzzahligen Variable `i` betrachte nur `i` gerade/ungerade
    - Die abstrakten Werte werden durch eine endliche Menge von *Prädikaten* beschrieben (dadurch wird der Zustandsraum endlich)
      - Programmzustände und -abläufe werden „approximiert“
      - Programme werden zu Booleschen Programmen (d.h. nur Boolesche Variablen)
        - Programme können mittels Model Checking analysiert werden
- **Anmerkung:** Programme kann man sich als Transitionssysteme vorstellen, definiert durch eine Transitionsrelation T(s,s’)

Prädikatabstraktion: Grundlagen

- **Konkrete Zustände:** Programmzustände
- **Abstrakte Zustände:** Fassen konkrete Programmzustände zusammen



Prädikatabstraktion: Grundlagen

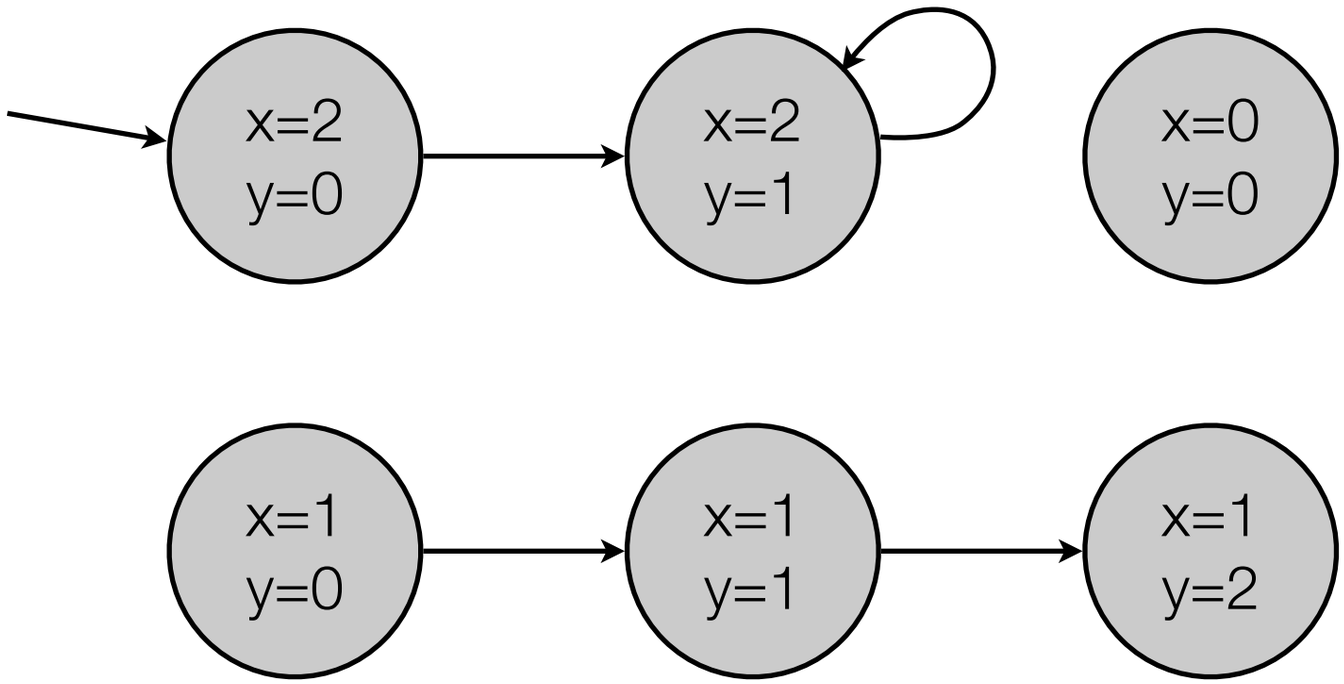
- **Konkrete Zustände:** Programmzustände
- **Abstrakte Zustände:** Fassen konkrete Programmzustände zusammen

$$S \underset{\gamma}{\overset{\alpha}{\rightleftarrows}} \hat{S}$$

- Wie kann eine Abstraktion definiert werden?
  - ➜ Durch Bewertungsfunktion auf den Prädikaten
  - Menge $\prod_1,...,\prod_n$ von Prädikaten über S
  - Abstrakte Zustände: $\hat{S} = \mathbb{B}^n$ (Wahrheitswerte der n Prädikate)
  - Abstraktionsfunktion: $\alpha(s) = \langle \prod_1(s), ..., \prod_n(s) \rangle$

Prädikatabstraktion: Beispiel

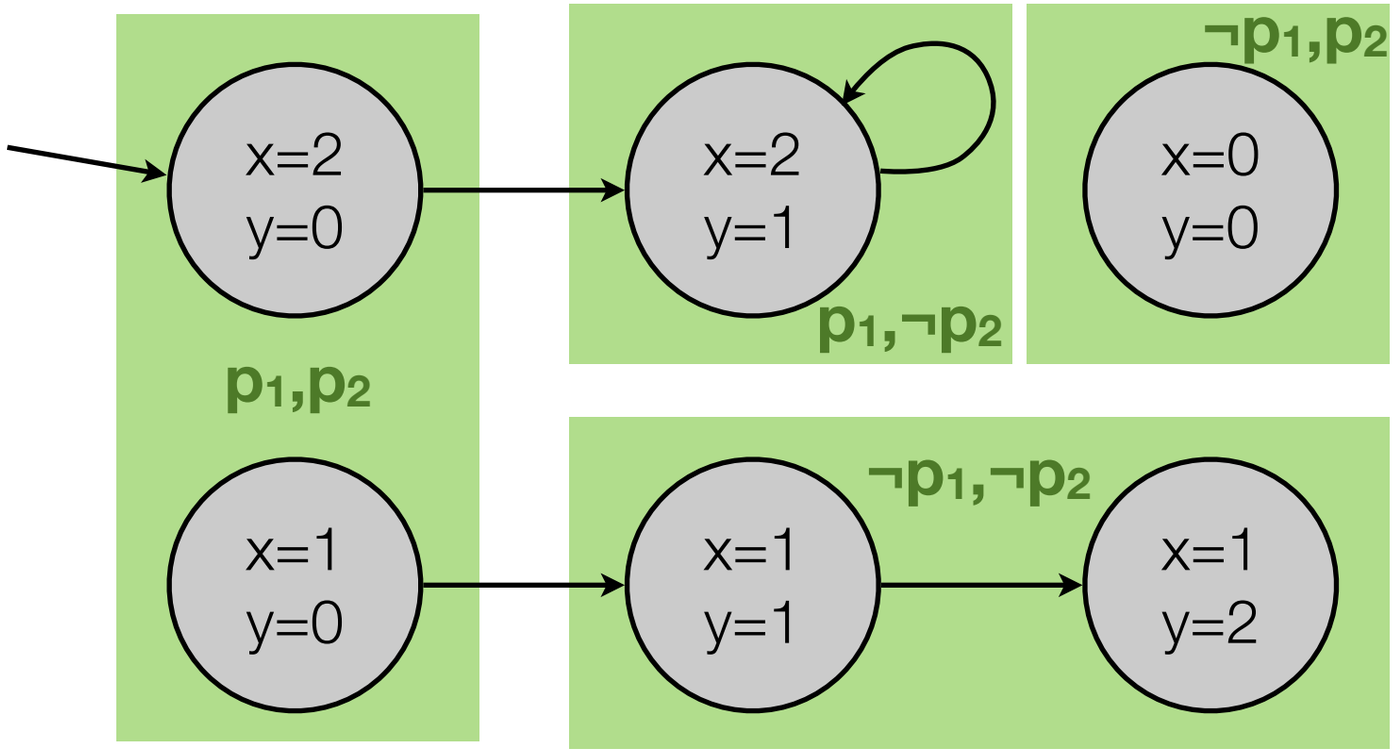
- Konkrete Zustände definiert durch Belegung zweier ganzzahliger Variablen (x, y)



- **Prädikate:**
  - $\prod_1(\langle x,y \rangle) = p_1 = (x > y)$
  - $\prod_2(\langle x,y \rangle) = p_2 = (y = 0)$

Prädikatabstraktion: Beispiel

- Konkrete Zustände definiert durch Belegung zweier ganzzahliger Variablen (x, y)

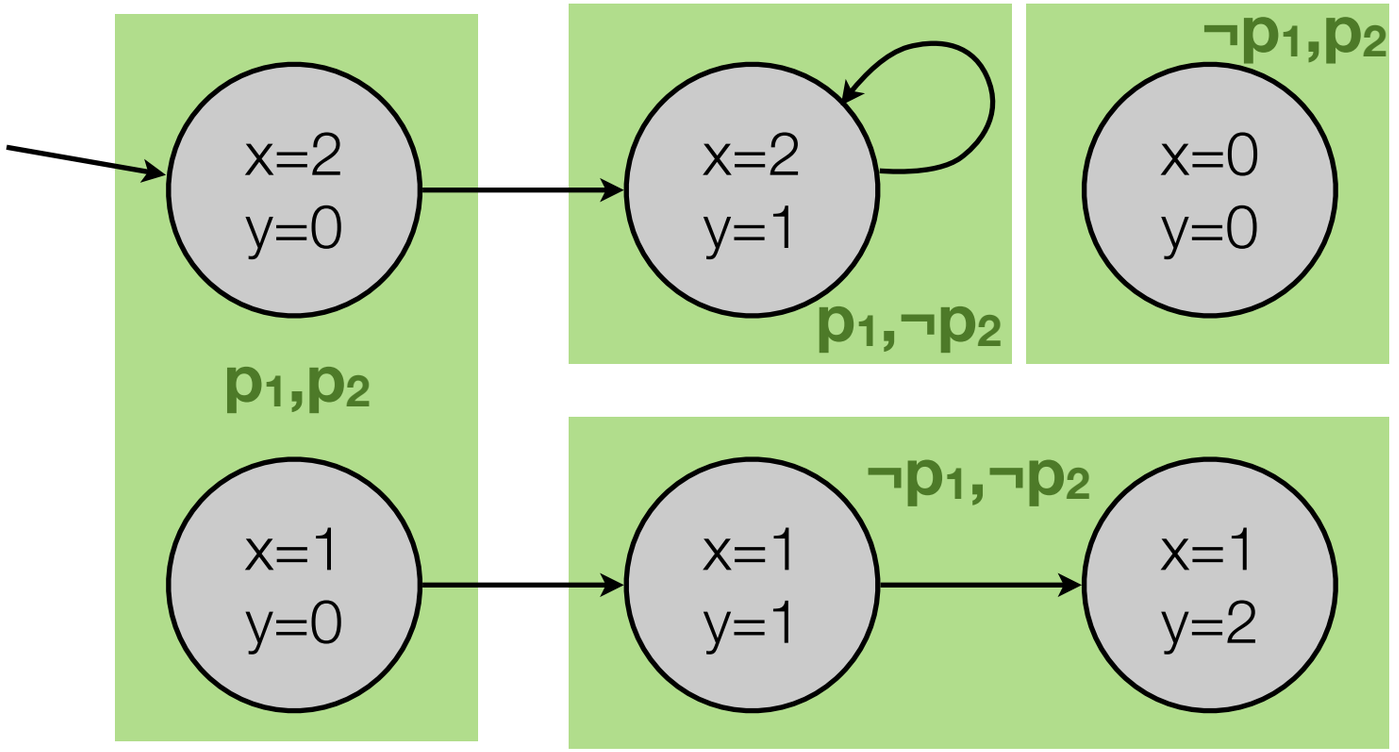


- **Prädikate:**
  - $\prod_1(\langle x,y \rangle) = p_1 = (x > y)$
  - $\prod_2(\langle x,y \rangle) = p_2 = (y = 0)$
- **Problem:** Wie berechnen wir die abstrakte Transitionsrelation?

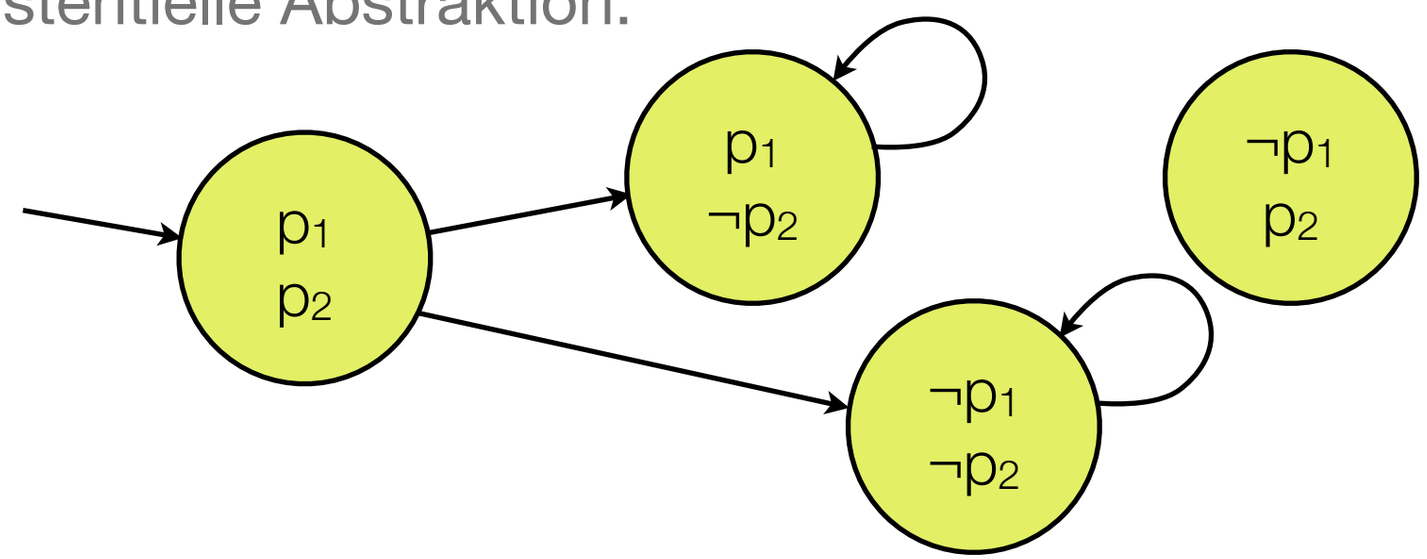
Existentielle Abstraktion

- **Gegeben:** (Konkretes) Modell $M = (S,S_0,T)$
- **Definition:** Ein Modell $\hat{M}=(\hat{S},\hat{S}_0,\hat{T})$ ist eine *existentielle Abstraktion* von $M = (S,S_0,T)$ bezüglich $\alpha: S \to \hat{S}$ genau dann, wenn
  - $\exists s \in S_0 \,.\, \alpha(s) = \hat{s} \;\Rightarrow\; \hat{s} \in \hat{S}_0$ und
  - $\exists (s,s') \in T \,.\, \alpha(s) = \hat{s} \wedge \alpha(s') = \hat{s}' \;\Rightarrow\; (\hat{s}, \hat{s}') \in \hat{T}$
- Für ein gegebenes α gibt es typischerweise eine Vielzahl von existentiellen Abstraktionen
- **Definition:** Ein Modell $M=(\hat{S},\hat{S}_0,\hat{T})$ ist eine *minimale existentielle Abstraktion* von $M = (S,S_0,T)$ bezüglich $\alpha: S \to \hat{S}$ genau dann, wenn
  - $\exists s \in S_0 \,.\, \alpha(s) = \hat{s} \;\Leftrightarrow\; \hat{s} \in \hat{S}_0$ und
  - $\exists (s,s') \in T \,.\, \alpha(s) = \hat{s} \wedge \alpha(s') = \hat{s}' \;\Leftrightarrow\; (\hat{s}, \hat{s}') \in \hat{T}$
- Die minimale existentielle Abstraktion ist die präziseste existentielle Abstraktion

Existentielle Abstraktion am Beispiel

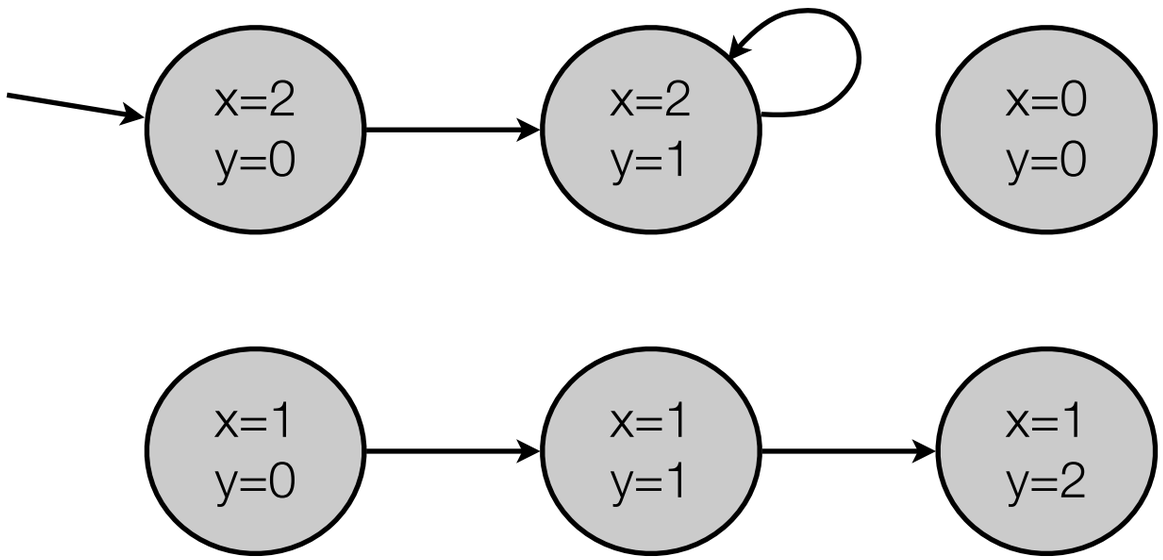


- Minimale existentielle Abstraktion:

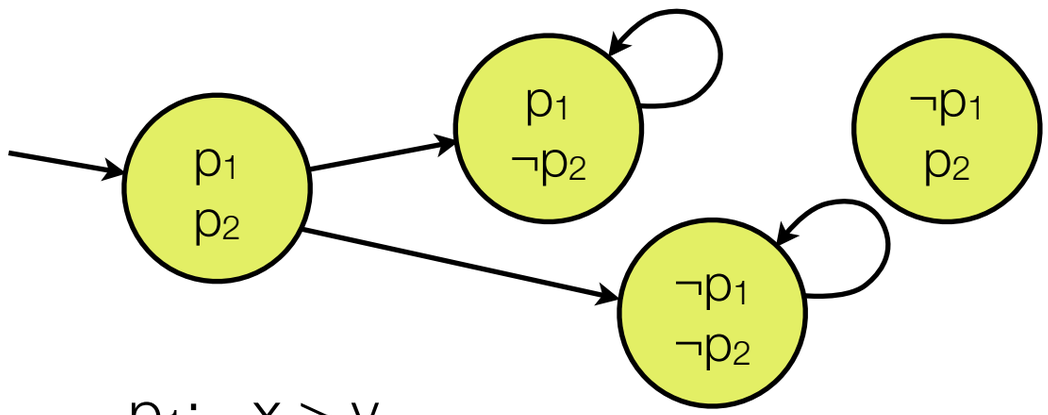


Existentielle Abstraktion am Beispiel

Konkret:



Abstrakt:



p_1: $x > y$
p_2: $y = 0$

- Wir wollen prüfen, ob für alle Abläufe unseres konkreten Programms eine Eigenschaft (global) gilt:
  1. $x > y \ \vee \ y \neq 0 \quad (= p_1 \vee \neg p_2)$
  2. $x > y \quad (= p_1)$
- Die 1. Eigenschaft gilt, die 2. aber nicht! („*spurious counterexample*“)

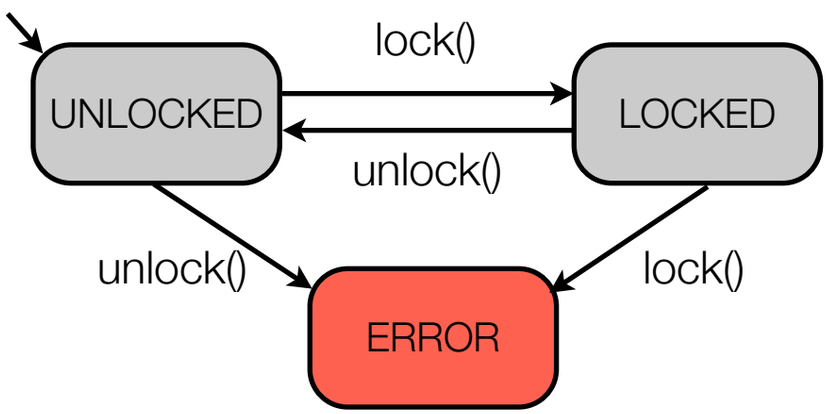
  ➜ Die Abstraktion war nicht passend und muss verändert werden („refinement“)

Refinement: Beispiel

```
LOCK = 0;
do {
  lock();
  nPacketsOld = nPackets;
  if (request) {
    request = request->next;
    unlock();
    nPackets++;
  }
} while (nPackets != nPacketsOld);
unlock();
```

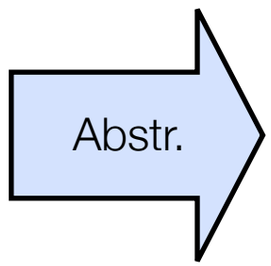
```
lock() {
  if (LOCK == 0)
    LOCK = 1;
  else ERROR();
}

unlock() {
  if (LOCK == 1)
    LOCK = 0;
  else ERROR();
}
```

- Arbeitet das Locking korrekt, d.h. ist ERROR() nicht erreichbar?



Refinement: Beispiel

```
LOCK = 0;
do {
  lock();
  nPacketsOld = nPackets;
  if (request) {
    request = request->next;
    unlock();
    nPackets++;
  }
} while (nPackets != nPacketsOld);
unlock();
```

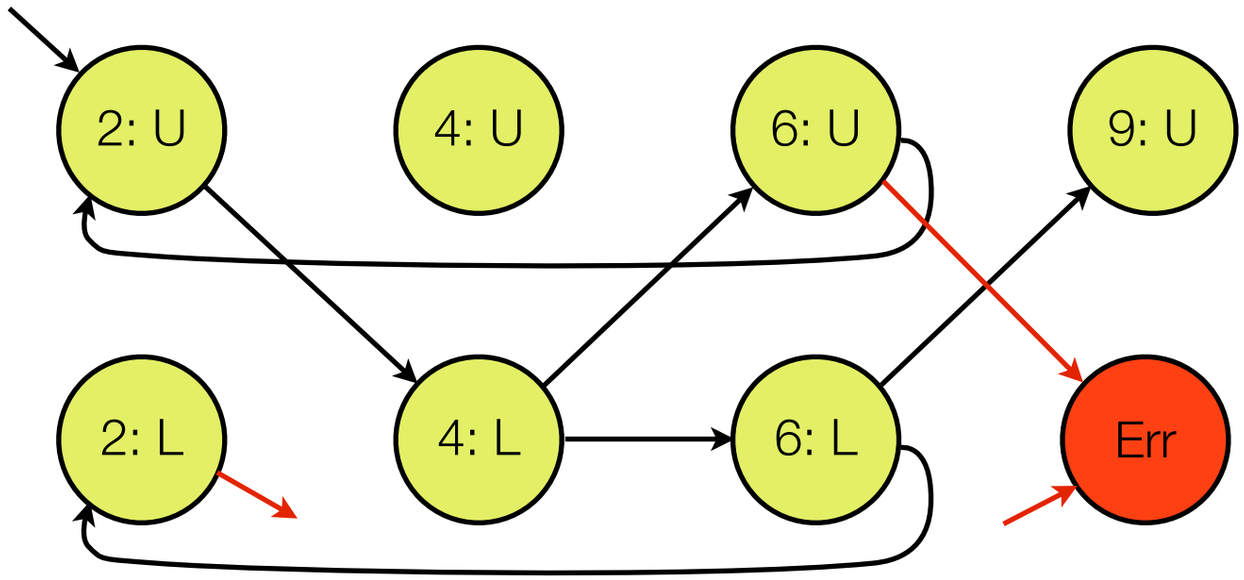



```
1:  LOCK = 0;
2:  do {
3:    lock();

4:  if (*) {

5:    unlock();

6:  }
7: } while (*);
8: unlock();
```

- Prädikat für Abstraktion:
  U: LOCK = 0
  (wir schreiben L für Zustände, in denen ¬U gilt)

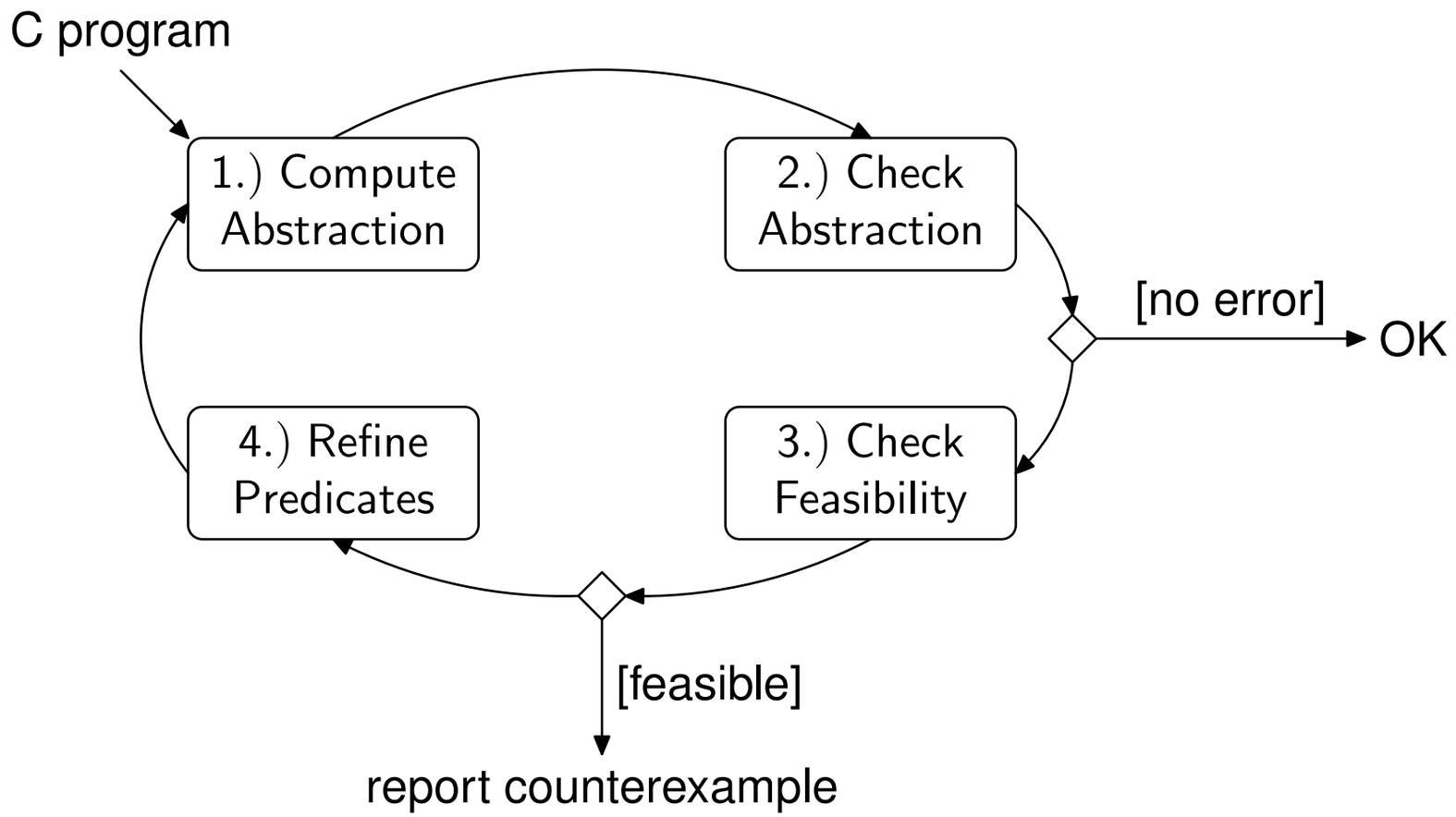
Refinement: Beispiel

```
1:  LOCK = 0;
2:  do {
3:    lock();
4:    if (*) {
5:      unlock();
6:    }
7:  } while (*);
8:  unlock();
9:
```



- Prädikat für Abstraktion:
  U: LOCK = 0
  (wir schreiben L für Zustände, in denen ¬U gilt)
- Ist der Pfad 2:U → 4:L → 6:U → Err auch im konkreten Programm möglich?

Abstraction-Refinement-Loop

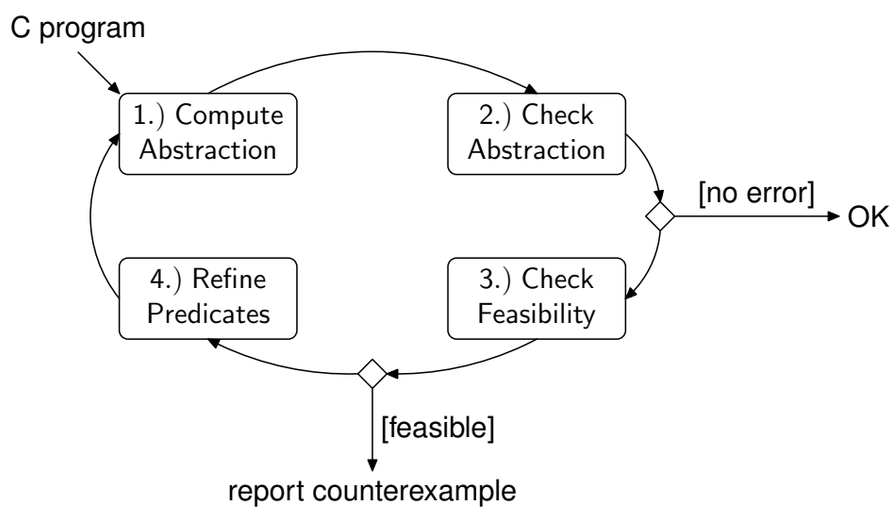


CEGAR: Counterexample-guided abstraction refinement

Welche Aufgaben sind zu bewerkstelligen?

1. Wie kann das abstrakte Transitionssystem berechnet werden?
2. Wie prüfen wir die Abstraktion?

   ➜ Model-Checking auf dem abstrakten Modell
3. Wie prüfen wir, ob ein abstrakter Pfad auch im konkreten Programm möglich ist?



4. Wie verfeinern wir die Abstraktion?

   Und wie wählen wir die erste Abstraktion?

SATABS: Berechnung abstraktes Transitionssystem

- Bringe Programm in SSA-Form
- Für jedes Prädikat π_i führe eine Boolesche Variable b_i ein.
- Jeder *basic block* entspricht einem Zustandsübergang
  - Codierung der *basic blocks*:
    - Details siehe Paper *Predicate Abstraction of ANSI-C Programs Using SAT (Clarke/Kroening/Sharygina/Yorav)*
    - Transitionssystem:

$$\Gamma(\bar{b}, \bar{b}', \bar{v}, \bar{v}') = (\bar{\pi}(\bar{v}) = \bar{b}) \wedge \mathcal{T}(\bar{v}, \bar{v}') \wedge (\bar{\pi}(\bar{v}') = \bar{b}')$$
$$\mathcal{B}(\bar{b}, \bar{b}') = \exists \bar{v}, \bar{v}' : \Gamma(\bar{b}, \bar{b}', \bar{v}, \bar{v}')$$

    - T(v,v') durch „bit-blasting“
- Control-flow-statements:
  - Falls Bedingung c bereits in π_i, verwende dieses.
  - Ansonsten: Berechne Zustände, in denen c gilt mittels SAT-Solver

SATABS: Simulation des abstrakten Gegenbeispiels

- Symbolische Ausführung:
  - T(v,v') des konkreten Programms werden anhand des CFG aus dem abstrakten Gegenbeispiel für die durchlaufenen *basic blocks* konkateniert
  - In jedem Schritt wird die Erfüllbarkeit mittels SAT-Solver geprüft
  - Wenn Formel unerfüllbar wird, kann der abstrakte Pfad nicht konkretisiert werden

    ➜ spurious counterexample
  - Kann der gesamte abstrakte Pfad nachvollzogen werden, so ist das Gegenbeispiel echt

SATABS: Verfeinerung der Abstraktion

- Verfeinerung wird durch hinzufügen weiterer Prädikate π erreicht
- Wie wird das neu hinzuzufügende Prädikat ausgewählt?
  - Betrachte Symbolische Ausführung des konkreten Gegenbeispiels, und dort die erste Stelle, an der es nicht weiter fortgesetzt werden kann
  - Die Prädikate an dieser Stelle sind Kandidaten zur Verfeinerung
- Typischerweise werden die Prädikate noch anhand des Unerfüllbarkeits-Beweises gefiltert