Karlsruhe Institute of Technology

Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

I: Einführung

Carsten Sinz
Institut für Theoretische Informatik

22.10.2019

Ist mein Programm korrekt?

Ist mein Programm korrekt?

- **Beispiel:** Bestimme Anzahl der 1-Bits in einem Wort (*population-count*)

Ist mein Programm korrekt?

- **Beispiel:** Bestimme Anzahl der 1-Bits in einem Wort (*population-count*)

```
uint32_t reference_popcount(uint32_t x)
{
    int i, s = 0;
    for(i = 0; i < 32; ++i)
        if(x & (1 << i))
            s++;
    return s;
}
```

Ist mein Programm korrekt?

- **Beispiel:** Bestimme Anzahl der 1-Bits in einem Wort (*population-count*)

```
uint32_t reference_popcount(uint32_t x)
{
    int i, s = 0;
    for(i = 0; i < 32; ++i)
        if(x & (1 << i))
            s++;
    return s;
}
```

```
uint32_t optimized_popcount(uint32_t x)
{
    x = x - ((x >> 1 ) & 0x55555555);
    x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
    x = (x + (x >> 4)) & 0x0F0F0F0F;
    x += (x >> 8);
    x += (x >> 16);
    return x & 0x0000003F;
}
```

Ist mein Programm korrekt?

- **Beispiel:** Bestimme Anzahl der 1-Bits in einem Wort (*population-count*)

```
uint32_t reference_popcount(uint32_t x)
{
    int i, s = 0;
    for(i = 0; i < 32; ++i)
        if(x & (1 << i))
            s++;
    return s;
}
```

```
uint32_t optimized_popcount(uint32_t x)
{
    x = x - ((x >> 1 ) & 0x55555555);
    x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
    x = (x + (x >> 4)) & 0x0F0F0F0F;
    x += (x >> 8);
    x += (x >> 16);
    return x & 0x0000003F;
}
```

- **Frage:** Liefert die optimierte Version immer das gleiche Ergebnis wie die Referenzimplementierung?

Der Logik-Ansatz

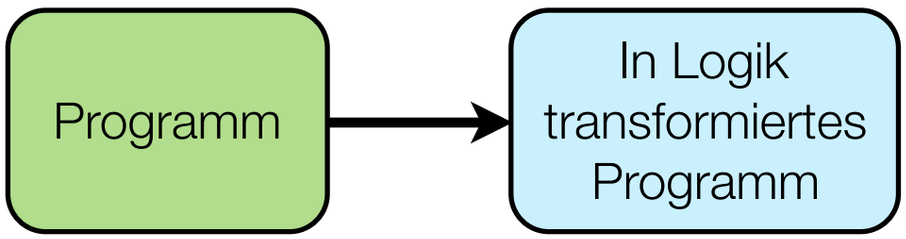
Der Logik-Ansatz

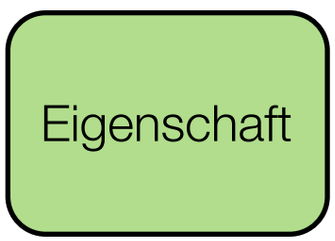
Der Logik-Ansatz

Programm

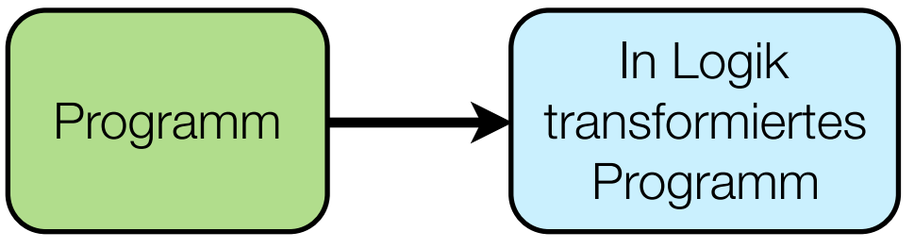
Eigenschaft

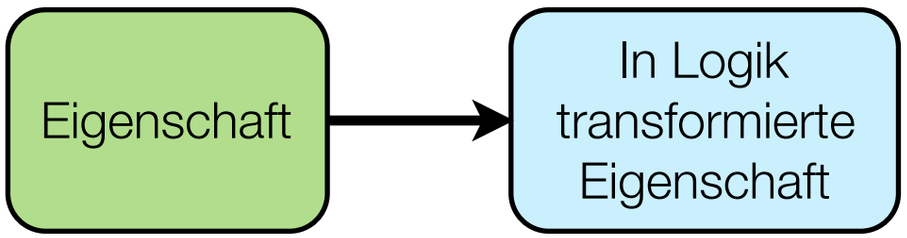
Der Logik-Ansatz



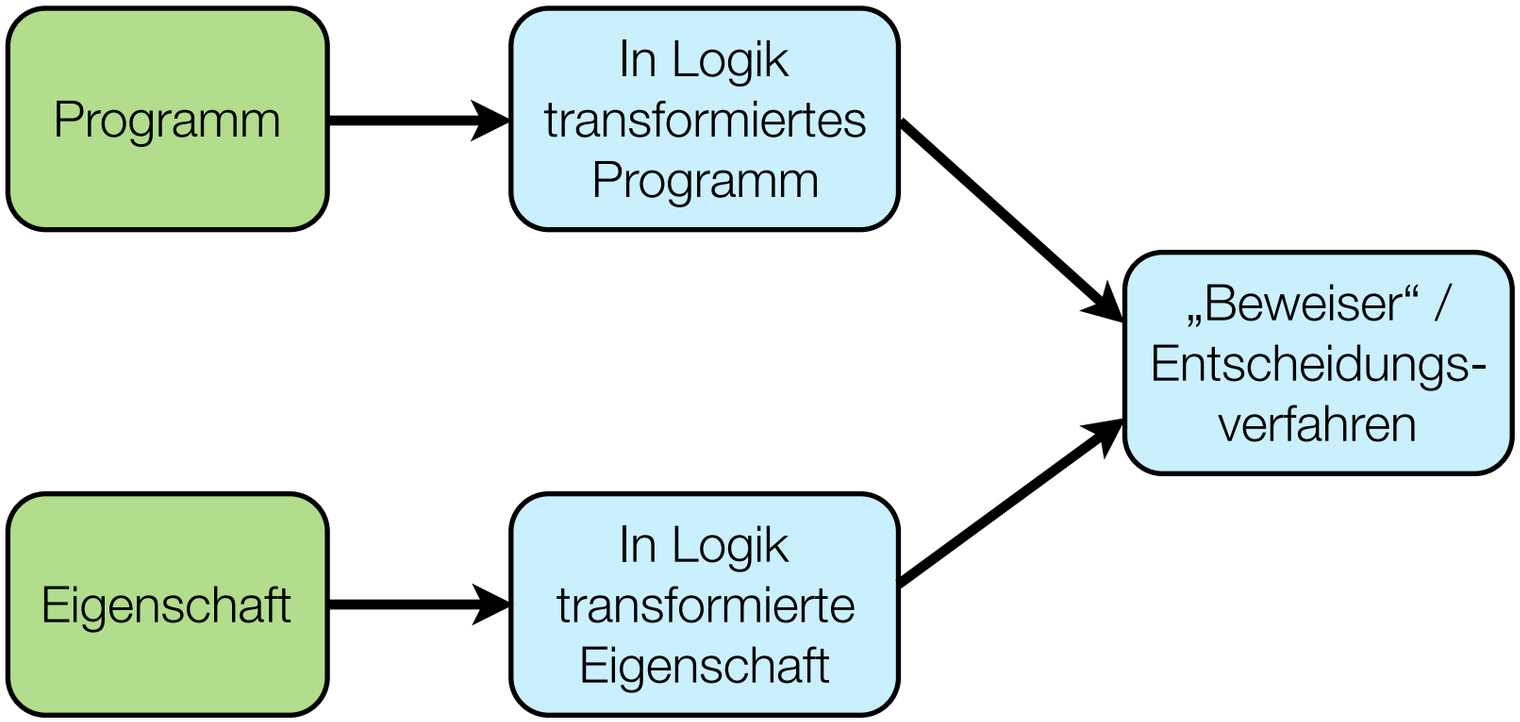


Der Logik-Ansatz

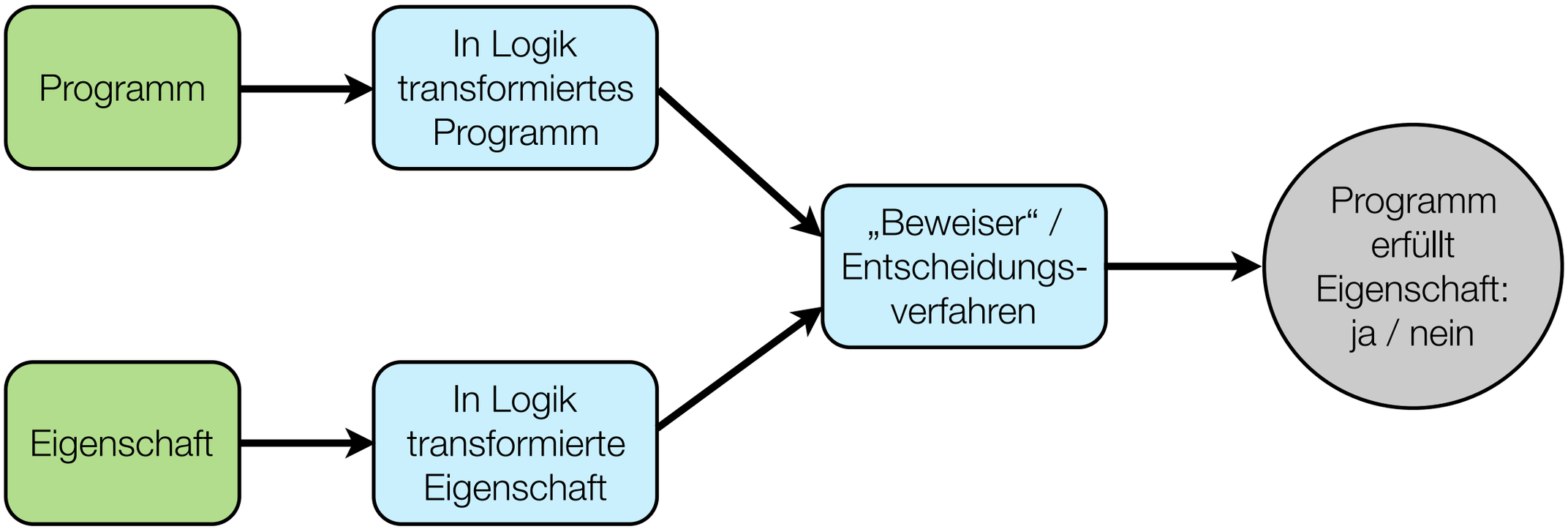




Der Logik-Ansatz



Der Logik-Ansatz



Zu klären

Zu klären

- **Welche Programme wollen wir prüfen?**

Zu klären

- **Welche Programme wollen wir prüfen?**

  ➜ C, C++, Java, ... (Imperative Programmiersprachen)

Zu klären

- **Welche Programme wollen wir prüfen?**

  ➜ C, C++, Java, ... (Imperative Programmiersprachen)

- **Welche Eigenschaften wollen wir prüfen?**

Zu klären

- **Welche Programme wollen wir prüfen?**

  ➜ C, C++, Java, ... (Imperative Programmiersprachen)

- **Welche Eigenschaften wollen wir prüfen?**

  ➜ Keine Programmabstürze

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse
  - ➜ Keine Sicherheitslücken

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse
  - ➜ Keine Sicherheitslücken
- **Welche Logik ist dafür geeignet?**

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse
  - ➜ Keine Sicherheitslücken

- **Welche Logik ist dafür geeignet?**
  - Modellierung von primitiven Datentypen (`int`, `float`, ...) und Operationen darauf

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse
  - ➜ Keine Sicherheitslücken

- **Welche Logik ist dafür geeignet?**
  - Modellierung von primitiven Datentypen (`int`, `float`, ...) und Operationen darauf
  - Modellierung von komplexen Datentypen

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse
  - ➜ Keine Sicherheitslücken

- **Welche Logik ist dafür geeignet?**
  - Modellierung von primitiven Datentypen (`int`, `float`, ...) und Operationen darauf
  - Modellierung von komplexen Datentypen
  - Modellierung von Programmfluss (Daten- und Kontrollfluss)

Zu klären

- **Welche Programme wollen wir prüfen?**
  - ➜ C, C++, Java, ... (Imperative Programmiersprachen)
- **Welche Eigenschaften wollen wir prüfen?**
  - ➜ Keine Programmabstürze
  - ➜ Keine „falschen“ Rechenergebnisse
  - ➜ Keine Sicherheitslücken

- **Welche Logik ist dafür geeignet?**
  - Modellierung von primitiven Datentypen (`int`, `float`, ...) und Operationen darauf
  - Modellierung von komplexen Datentypen
  - Modellierung von Programmfluss (Daten- und Kontrollfluss)
  - Modellierung von Speicher

Welche Logik?

Welche Logik?

- **Erwünschte Eigenschaften:**

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**
  - Aussagenlogik

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**
  - Aussagenlogik
  - Prädikatenlogik erster Stufe

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**
  - Aussagenlogik
  - Prädikatenlogik erster Stufe
  - CTL / LTL

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**
  - Aussagenlogik
  - Prädikatenlogik erster Stufe
  - CTL / LTL
  - Spezielle Logiken

Welche Logik?

- **Erwünschte Eigenschaften:**
  - Möglichst ausdrucksstark
  - Möglichst entscheidbar
  - Eingebaute Theorien (z.B. Arithmetik)
- **Kandidaten:**
  - Aussagenlogik
  - Prädikatenlogik erster Stufe
  - CTL / LTL
  - Spezielle Logiken

Definition Entscheidbarkeit

Definition Entscheidbarkeit

- **Definition (Erfüllbarkeit, *Gültigkeit*):**
  Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.

Definition Entscheidbarkeit

- **Definition (Erfüllbarkeit, *Gültigkeit*):**
  Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.
- **Definition (*Entscheidungsproblem*):**
  Das *Entscheidungsproblem* für eine gegebene Formel (in einer Logik L) ϕ ist zu bestimmen, ob ϕ gültig ist.

Definition Entscheidbarkeit

- **Definition (Erfüllbarkeit, *Gültigkeit*):**
  Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.

- **Definition (*Entscheidungsproblem*):**
  Das *Entscheidungsproblem* für eine gegebene Formel (in einer Logik L) ϕ ist zu bestimmen, ob ϕ gültig ist.

- **Definition (*Korrektheit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *korrekt*, falls immer wenn er „gültig“ als Ergebnis liefert die Eingabeformel wirklich gültig ist.

Definition Entscheidbarkeit

- **Definition (Erfüllbarkeit, *Gültigkeit*):**
  Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.

- **Definition (*Entscheidungsproblem*):**
  Das *Entscheidungsproblem* für eine gegebene Formel (in einer Logik L) ϕ ist zu bestimmen, ob ϕ gültig ist.

- **Definition (*Korrektheit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *korrekt*, falls immer wenn er „gültig“ als Ergebnis liefert die Eingabeformel wirklich gültig ist.

- **Definition (*Vollständigkeit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *vollständig*, wenn er (a) immer terminiert und (b) „gültig“ als Ergebnis liefert, sofern die Eingabeformel gültig ist.

Definition Entscheidbarkeit

- **Definition (Erfüllbarkeit, *Gültigkeit*):**
  Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.

- **Definition (*Entscheidungsproblem*):**
  Das *Entscheidungsproblem* für eine gegebene Formel (in einer Logik L) ϕ ist zu bestimmen, ob ϕ gültig ist.

- **Definition (*Korrektheit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *korrekt*, falls immer wenn er „gültig“ als Ergebnis liefert die Eingabeformel wirklich gültig ist.

- **Definition (*Vollständigkeit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *vollständig*, wenn er (a) immer terminiert und (b) „gültig“ als Ergebnis liefert, sofern die Eingabeformel gültig ist.

- **Definition (*Entscheidungsverfahren*):**
  Ein Algorithmus A heißt *Entscheidungsverfahren* für eine Logik L, wenn A korrekt und vollständig ist für jede Formel aus L.

Definition Entscheidbarkeit

- **Definition (Erfüllbarkeit, *Gültigkeit*):**
  Eine Formel ist *erfüllbar*, wenn es *eine* Variablenbelegung und eine Interpretation der (nicht-logischen) Symbole *gibt*, so dass die Formel zu **wahr** evaluiert. Eine Formel ist gültig, wenn sie für *alle* Variablenbelegungen und *alle* Interpretationen der (nicht-logischen) Symbole zu **wahr** evaluiert.

- **Definition (*Entscheidungsproblem*):**
  Das *Entscheidungsproblem* für eine gegebene Formel (in einer Logik L) ϕ ist zu bestimmen, ob ϕ gültig ist.

- **Definition (*Korrektheit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *korrekt*, falls immer wenn er „gültig“ als Ergebnis liefert die Eingabeformel wirklich gültig ist.

- **Definition (*Vollständigkeit*):**
  Ein Algorithmus für das Entscheidungsproblem ist *vollständig*, wenn er (a) immer terminiert und (b) „gültig“ als Ergebnis liefert, sofern die Eingabeformel gültig ist.

- **Definition (*Entscheidungsverfahren*):**
  Ein Algorithmus A heißt *Entscheidungsverfahren* für eine Logik L, wenn A korrekt und vollständig ist für jede Formel aus L.

- **Definition (*Entscheidbarkeit einer Logik*):**
  Eine Logik L heißt entscheidbar, wenn es ein Entscheidungsverfahren für L gibt.

Entscheidbarkeit für verschiedene Logiken

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit

Entscheidbarkeit für verschiedene Logiken

Logik	**Entscheidbarkeit**
Prädikatenlogik (erster Stufe)	

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**
Polynomgleichungen über den ganzen Zahlen	

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**
Polynomgleichungen über den ganzen Zahlen	**unentscheidbar**

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**
Polynomgleichungen über den ganzen Zahlen	**unentscheidbar**
Presburger-Arithmetik	

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**
Polynomgleichungen über den ganzen Zahlen	**unentscheidbar**
Presburger-Arithmetik	**entscheidbar**

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**
Polynomgleichungen über den ganzen Zahlen	**unentscheidbar**
Presburger-Arithmetik	**entscheidbar**
Quantorenfreie Logik der Bitvektoren	

Entscheidbarkeit für verschiedene Logiken

Logik	Entscheidbarkeit
Prädikatenlogik (erster Stufe)	**unentscheidbar**
Monadische Prädikatenlogik	**entscheidbar**
Polynomgleichungen über den ganzen Zahlen	**unentscheidbar**
Presburger-Arithmetik	**entscheidbar**
Quantorenfreie Logik der Bitvektoren	**entscheidbar**

Beispiele für einfach Entscheidungsverfahren

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?
    - Hängt von Grundbereich ab, über dem wir die Formel interpretieren!

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?
    - Hängt von Grundbereich ab, über dem wir die Formel interpretieren!
    - Über $\mathbb{R}$: Lösbar (d.h. erfüllbar), wenn $b^2 - 4ac \geq 0$.

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?
    - Hängt von Grundbereich ab, über dem wir die Formel interpretieren!
    - Über $\mathbb{R}$: Lösbar (d.h. erfüllbar), wenn $b^2 - 4ac \geq 0$.
    - Über $\mathbb{C}$: Immer lösbar.

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?
    - Hängt von Grundbereich ab, über dem wir die Formel interpretieren!
    - Über $\mathbb{R}$: Lösbar (d.h. erfüllbar), wenn $b^2 - 4ac \geq 0$.
    - Über $\mathbb{C}$: Immer lösbar.
    - Über $\mathbb{Q}$? Über $\mathbb{Z}$? Über algebraischen Körpererweiterungen von $\mathbb{Q}$?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?
    - Hängt von Grundbereich ab, über dem wir die Formel interpretieren!
    - Über $\mathbb{R}$: Lösbar (d.h. erfüllbar), wenn $b^2 - 4ac \geq 0$.
    - Über $\mathbb{C}$: Immer lösbar.
    - Über $\mathbb{Q}$? Über $\mathbb{Z}$? Über algebraischen Körpererweiterungen von $\mathbb{Q}$?
  - Entscheidungsverfahren / Algorithmus (zumindest für $\mathbb{R}, \mathbb{C}$) sehr einfach

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 1: Lösen quadratischer Gleichungen**
  - Wann besitzt die Gleichung $ax^2 + bx + c = 0$ eine Lösung / Lösungen?
  - Wir wissen: $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ beschreibt die Lösungen
  - Lösbar?
    - Hängt von Grundbereich ab, über dem wir die Formel interpretieren!
    - Über $\mathbb{R}$: Lösbar (d.h. erfüllbar), wenn $b^2 - 4ac \geq 0$.
    - Über $\mathbb{C}$: Immer lösbar.
    - Über $\mathbb{Q}$? Über $\mathbb{Z}$? Über algebraischen Körpererweiterungen von $\mathbb{Q}$?
  - Entscheidungsverfahren / Algorithmus (zumindest für $\mathbb{R}, \mathbb{C}$) sehr einfach
  - Wir sind häufig nicht nur an der Lösbarkeit interessiert, sondern auch an einer oder sogar allen Lösungen.

Beispiele für einfach Entscheidungsverfahren

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**
  - Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \quad \vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

  eine Lösung / Lösungen?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**
  - Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \quad \vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

    eine Lösung / Lösungen?
  - Wie sieht ein Lösungsverfahren aus?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**
  - Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \quad \vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

    eine Lösung / Lösungen?
  - Wie sieht ein Lösungsverfahren aus?
  - Gaußsches Eliminationsverfahren (für $\mathbb{Q}$)

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**
  - Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \quad \vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

    eine Lösung / Lösungen?
  - Wie sieht ein Lösungsverfahren aus?
  - Gaußsches Eliminationsverfahren (für $\mathbb{Q}$)
  - Funktioniert dies auch über $\mathbb{Z}$?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**
  - Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \quad \vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

    eine Lösung / Lösungen?
  - Wie sieht ein Lösungsverfahren aus?
  - Gaußsches Eliminationsverfahren (für $\mathbb{Q}$)
  - Funktioniert dies auch über $\mathbb{Z}$?
  - Komplexität?

Beispiele für einfach Entscheidungsverfahren

- **Beispiel 2: Lineare Gleichungssysteme**
  - Wann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{n,1}x_n + a_{n-1,1}x_{n-1} + \cdots + a_{1,1}x_1 &= b_1 \\ &\vdots \quad \vdots \\ a_{n,m}x_n + a_{n-1,m}x_{n-1} + \cdots + a_{1,m}x_1 &= b_m \end{aligned}$$

    eine Lösung / Lösungen?
  - Wie sieht ein Lösungsverfahren aus?
  - Gaußsches Eliminationsverfahren (für $\mathbb{Q}$)
  - Funktioniert dies auch über $\mathbb{Z}$?
  - Komplexität?
    - Was ist eine elementare Operation? Koeffizienten können möglicherweise sehr groß werden ➜ Bit-Komplexität

Beobachtungen

- **Entscheidungsverfahren sind abhängig von:**
  - Erlaubten Funktions- und Relationssymbolen ($\mathbf{F}$, $\mathbf{R}$)
  - Grundbereich, über dem eine Formel interpretiert wird ($\mathbf{U}$)
  - Axiomen ($\mathbf{A}$), die gelten sollen
- **Satisfiability Modulo Theory (SMT)**
  - SMT-Solver implementieren Entscheidungsverfahren für gegebene $\mathbf{F}$, $\mathbf{R}$, $\mathbf{U}$ und $\mathbf{A}$ (d.h., Axiome und Grundbereich sind „eingebaut“).
  - Standard für SMT-Theorien: SMT-LIB
- **Komplexität in Praxis wichtig**
  - für arithmetische Theorien häufig exponentiell

Vorlesungsüberblick / Themen

Theorie	Einführung
	Theoretische Grundlagen, Beispiele
	SAT-Solving: Fortgeschrittene Algorithmen
	DPLL(T): DPLL mit Theorie-Atomen
	Uninterpretierte Funktionen mit Gleichheit (EUF)
	Lineare Arithmetik der ganzen Zahlen (Fourier-Motzkin, Omega-Test)
	Logik der Bitvektoren
	Array-Logik
	Modulare Arithmetik
	Quantorenelimination für reell abgeschlossene Körper (CAD)
	Kombination von Theorien
Anwendung	Polyspace: Abstract Interpretation
	LLBMC: Software Bounded Model Checking
	KLEE: Symbolic Execution, SATABS: Predicate Abstraction
	Lab: Polyspace, LLBMC, KLEE, SATABS

Vorlesungsüberblick / Themen

Theorie	Einführung
	Theoretische Grundlagen, Beispiele
	SAT-Solving: Fortgeschrittene Algorithmen
	DPLL(T): DPLL mit Theorie-Atomen
	Uninterpretierte Funktionen mit Gleichheit (EUF)
	Lineare Arithmetik der ganzen Zahlen (Fourier-Motzkin, Omega-Test)
	Logik der Bitvektoren
	Array-Logik
	Modulare Arithmetik
	Quantorenelimination für reell abgeschlossene Körper (CAD)
	Kombination von Theorien
Anwendung	Polyspace: Abstract Interpretation
	LLBMC: Software Bounded Model Checking
	KLEE: Symbolic Execution, SATABS: Predicate Abstraction
	Lab: Polyspace, LLBMC, KLEE, SATABS

Übungen 14-tägig, erster Termin: 06.11.

Enthält dieser C-Code einen Fehler?

```
int SATD (void)
{
  int d[16];
   ⋮

  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

[Code from SPEC CPU 2006 Benchmark `464.h264ref`; via John Regehr's blog *Embedded in Academia*; March 22, 2013]

Enthält dieser C-Code einen Fehler?

```
int SATD (void)
{
  int d[16];
   ⋮

  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

array index out of bounds in iteration 16

[Code from SPEC CPU 2006 Benchmark `464.h264ref`; via John Regehr's blog *Embedded in Academia*; March 22, 2013]

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**

```
int SATD (void)
{
  int d[16];
   ⋮
  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**
  - What are the consequences?
  - How can we detect this error?
    By testing?

```
int SATD (void)
{
  int d[16];
   ⋮
  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**
  - What are the consequences?

    None?!
  - How can we detect this error?
    By testing?

```
int SATD (void)
{
  int d[16];
   ⋮
  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**
  - What are the consequences?
    None?!
  - How can we detect this error?
    By testing? **No!**

```
int SATD (void)
{
  int d[16];
   ⋮
  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**
  - What are the consequences?
    None?!
  - How can we detect this error?
    By testing? **No!**

```
int SATD (void)
{
  int d[16];
   ⋮

  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

GCC pre-4.8 generates the following code:

```
SATD:
     ...
.L2:
     jmp .L2
```

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**
  - What are the consequences?

    Unexpected behavior!
  - How can we detect this error?
    By testing?

```
int SATD (void)
{
  int d[16];
   ⋮
  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

GCC pre-4.8 generates the following code:

```
SATD:
     ...
.L2:
     jmp .L2
```

Enthält dieser C-Code einen Fehler?

- **Array-index-out-of-bounds error**
  - What are the consequences?

    Unexpected behavior!
  - How can we detect this error?
    By testing? **Perhaps (depending on the compiler)**

```
int SATD (void)
{
  int d[16];
   ⋮

  int satd = 0, dd, k;
  for (dd=d[k=0]; k<16; dd=d[++k]) {
    satd += (dd < 0 ? -dd : dd);
  }
  return satd;
}
```

GCC pre-4.8 generates the following code:

```
SATD:
     ...
.L2:
     jmp .L2
```

Literatur

- Daniel Kröning / Ofer Strichman:
  Decision Procedures – An Algorithmic Point of View
  (Springer, 2008)
  [Innerhalb des Uni-Netzes ist eine Online-Version im Volltext verfügbar.]

- Aaron R. Bradley / Zohar Manna:
  The Calculus of Computation: Decision Procedures with Applications to Verification
  (Springer, 2007)

- Weitere Literatur wird während der Vorlesung genannt