Angewandte Logik:

Von Tante Agatha zu korrekter Software

Viorica Sofronie-Stokkermans

Uwe Waldmann

MPI für Informatik

Logik in der Informatik

Was ist Logik?

Logik in der Informatik

Was ist Logik?

- Mathematisch?

Logik in der Informatik

Was ist Logik?

- Mathematisch?

- Unverständlich?

Logik in der Informatik

Was ist Logik?

- Mathematisch?

- Unverständlich?

- Reine Theorie ohne praktischen Nutzen?

Logik in der Informatik

Was ist Logik?

- Mathematisch?

  ja

- Unverständlich?

- Reine Theorie ohne praktischen Nutzen?

Logik in der Informatik

Was ist Logik?

- Mathematisch?

  ja

- Unverständlich?

  hoffentlich nein

- Reine Theorie ohne praktischen Nutzen?

Logik in der Informatik

Was ist Logik?

- Mathematisch?

  ja

- Unverständlich?

  hoffentlich nein

- Reine Theorie ohne praktischen Nutzen?

  nein: Verifikation von Hardware, Software, Protokollen;
  Sprachverarbeitung und Wissensrepräsentation;
  Abfragesprachen für Datenbanken; ...

Beispiel 1: Tante Agatha

Eine kleine Logelei:

> Jemand, der in Schloß Dreadbury wohnt, hat Tante Agatha ermordet. Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloß Dreadbury. Ein Mörder haßt immer sein Opfer und ist niemals reicher als sein Opfer. Charles haßt niemanden, den Tante Agatha gehaßt hat. Agatha hat jeden gehaßt außer ihrem Butler. Der Butler haßt jeden, der nicht reicher ist als Tante Agatha. Der Butler haßt jeden, den Tante Agatha gehaßt hat. Niemand haßt jeden. Agatha war nicht der Butler.

Wer hat Tante Agatha ermordet?

Beispiel 1: Tante Agatha

Bevor wir das Problem von einem automatischen Beweiser lösen lassen können, müssen wir es formalisieren:

Beispiel 1: Tante Agatha

Bevor wir das Problem von einem automatischen Beweiser lösen lassen können, müssen wir es formalisieren:

> Jemand, der in Schloß Dreadbury wohnt, hat Tante Agatha ermordet.

Beispiel 1: Tante Agatha

Bevor wir das Problem von einem automatischen Beweiser lösen lassen können, müssen wir es formalisieren:

Jemand, der in Schloß Dreadbury wohnt, hat Tante Agatha ermordet.

- $\exists x\ (\text{schloßbewohner}(x) \wedge \text{ermordet}(x, \text{a}))$

Beispiel 1: Tante Agatha

Bevor wir das Problem von einem automatischen Beweiser lösen lassen können, müssen wir es formalisieren:

Jemand, der in Schloß Dreadbury wohnt, hat Tante Agatha ermordet.

- $\exists x\ (\text{schloßbewohner}(x) \wedge \text{ermordet}(x, \text{a}))$

Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloß Dreadbury.

Beispiel 1: Tante Agatha

Bevor wir das Problem von einem automatischen Beweiser lösen lassen können, müssen wir es formalisieren:

Jemand, der in Schloß Dreadbury wohnt, hat Tante Agatha ermordet.

- $\exists x\ (\mathrm{schloßbewohner}(x) \wedge \mathrm{ermordet}(x, \mathrm{a}))$

Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloß Dreadbury.

- $\forall x\ (\mathrm{schloßbewohner}(x) \leftrightarrow (x = \mathrm{a} \vee x = \mathrm{b} \vee x = \mathrm{c}))$

Beispiel 1: Tante Agatha

Ein Mörder haßt immer sein Opfer und ist niemals reicher als sein Opfer.

- $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \text{haßt}(x, y))$
  $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \neg\, \text{reicher}(x, y))$

Charles haßt niemanden, den Tante Agatha gehaßt hat.

- $\forall x\ (\text{haßt}(\mathsf{c}, x) \rightarrow \neg\, \text{haßt}(\mathsf{a}, x))$

Agatha hat jeden gehaßt außer ihrem Butler.

- $\forall x\ (\neg\, \text{haßt}(\mathsf{a}, x) \leftrightarrow x = \mathsf{b})$

Beispiel 1: Tante Agatha

Der Butler haßt jeden, der nicht reicher ist als Tante Agatha.

- $\forall x\ (\neg\, \mathrm{reicher}(x, \mathrm{a}) \rightarrow \mathrm{haßt}(\mathrm{b}, x))$

Der Butler haßt jeden, den Tante Agatha gehaßt hat.

- $\forall x\ (\mathrm{haßt}(\mathrm{a}, x) \rightarrow \mathrm{haßt}(\mathrm{b}, x))$

Niemand haßt jeden.

- $\forall x\ \exists y\ (\neg\, \mathrm{haßt}(x, y))$

Agatha war nicht der Butler.

- $\neg\, \mathrm{a} = \mathrm{b}$

Beispiel 1: Tante Agatha

Nun können wir aus den Formeln neue Formeln ableiten.
Beispielsweise so:

$$\frac{\text{ermordet}(x, y) \rightarrow \text{haßt}(x, y) \qquad \text{haßt}(\text{c}, y) \rightarrow \neg\, \text{haßt}(\text{a}, y)}{\text{ermordet}(\text{c}, y) \rightarrow \neg\, \text{haßt}(\text{a}, y)}$$

Beispiel 1: Tante Agatha

Nun können wir aus den Formeln neue Formeln ableiten.
Beispielsweise so:

$$\frac{\mathrm{ermordet}(x, y) \rightarrow \mathrm{haßt}(x, y) \qquad \mathrm{haßt}(\mathrm{c}, y) \rightarrow \neg\, \mathrm{haßt}(\mathrm{a}, y)}{\mathrm{ermordet}(\mathrm{c}, y) \rightarrow \neg\, \mathrm{haßt}(\mathrm{a}, y)}$$

$$\frac{\mathrm{ermordet}(\mathrm{c}, y) \rightarrow \neg\, \mathrm{haßt}(\mathrm{a}, y) \qquad \neg\, \mathrm{haßt}(\mathrm{a}, y) \rightarrow y = \mathrm{b}}{\mathrm{ermordet}(\mathrm{c}, y) \rightarrow y = \mathrm{b}}$$

Beispiel 1: Tante Agatha

Nun können wir aus den Formeln neue Formeln ableiten.

Beispielsweise so:

$$\frac{\mathsf{ermordet}(x, y) \rightarrow \mathsf{haßt}(x, y) \qquad \mathsf{haßt}(\mathsf{c}, y) \rightarrow \neg\,\mathsf{haßt}(\mathsf{a}, y)}{\mathsf{ermordet}(\mathsf{c}, y) \rightarrow \neg\,\mathsf{haßt}(\mathsf{a}, y)}$$

$$\frac{\mathsf{ermordet}(\mathsf{c}, y) \rightarrow \neg\,\mathsf{haßt}(\mathsf{a}, y) \qquad \neg\,\mathsf{haßt}(\mathsf{a}, y) \rightarrow y = \mathsf{b}}{\mathsf{ermordet}(\mathsf{c}, y) \rightarrow y = \mathsf{b}}$$

$$\frac{\mathsf{ermordet}(\mathsf{c}, y) \rightarrow y = \mathsf{b} \qquad \neg\,\mathsf{a} = \mathsf{b}}{\neg\,\mathsf{ermordet}(\mathsf{c}, \mathsf{a})}$$

Beispiel 1: Tante Agatha

Damit wissen wir nun schon, daß Charles nicht der Mörder ist. Die weitere Berechnung ist aber mühsam.

Glücklicherweise können wir die Arbeit einem automatischen Theorembeweiser überlassen.

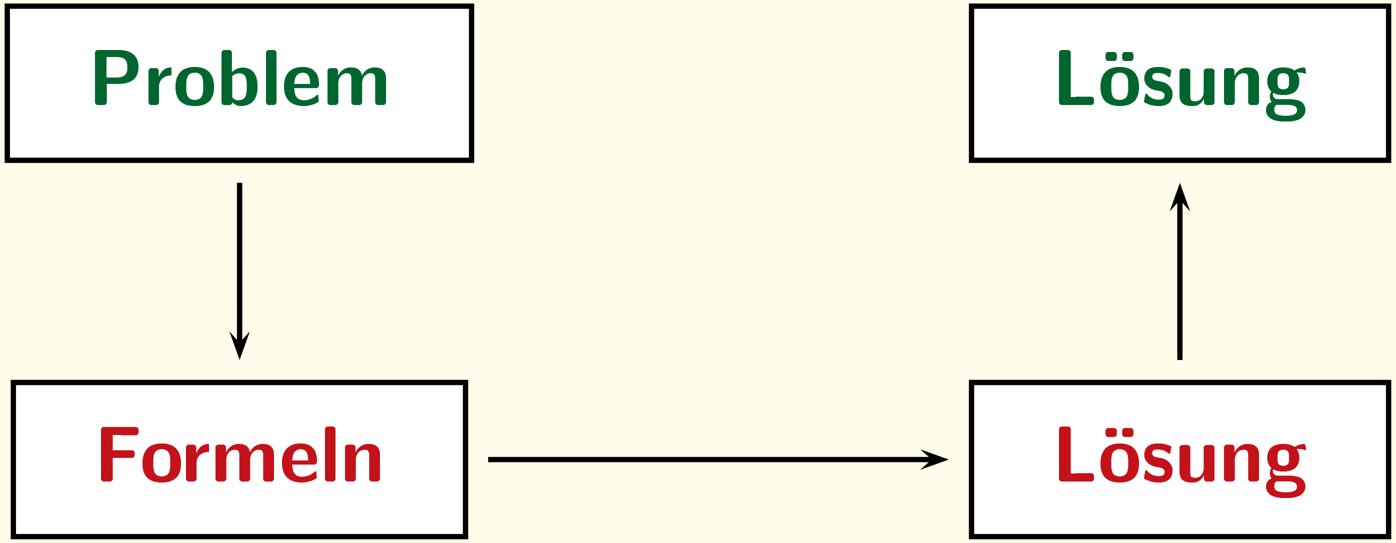
Das Prinzip

Problem - - - - - - - - → **Lösung**

Problem	Lösung
Beschreibung der Situation in Schloß Dreadbury	Charles ermordete Agatha
	Der Butler ermordete Agatha
	Agatha beging Selbstmord
Wer hat Agatha ermordet?	Täter unbekannt

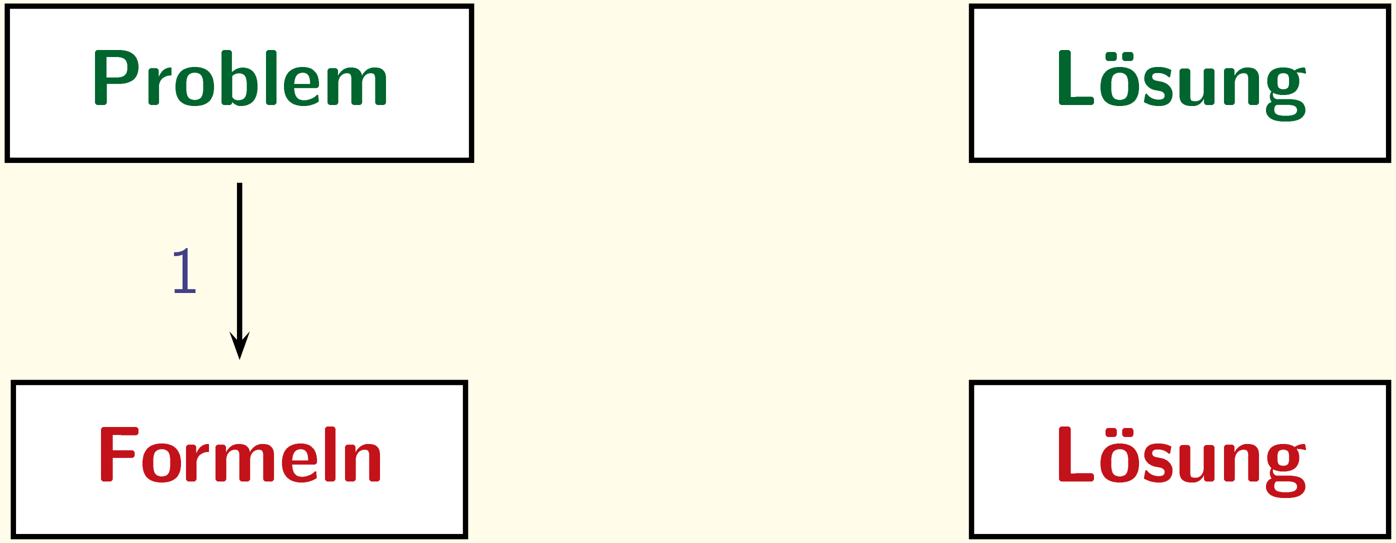
Das Prinzip

z. B. natürliche Sprache



präzise Beschreibung

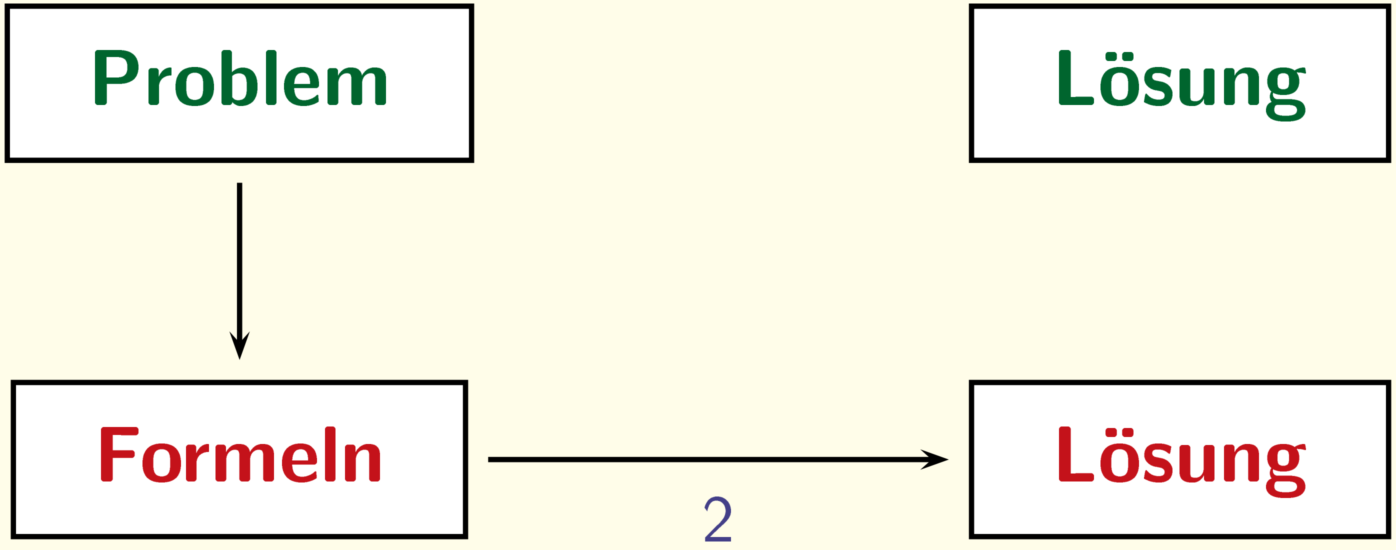
Das Prinzip



1. Formalisierung: von Problemen zu Formeln

   manchmal automatisch möglich

Das Prinzip



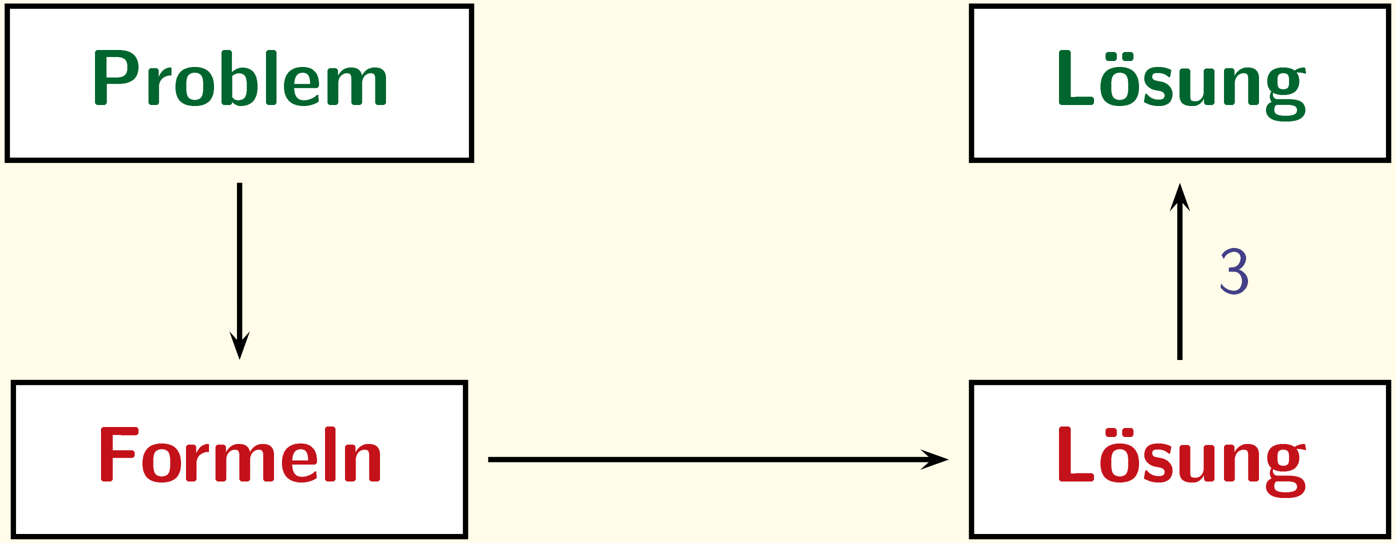
2. Formalisiertes Problem lösen

  in der Praxis entstehen **sehr** viele neue Formeln

  Computerunterstützung ist notwendig

  (aber auch dann ist die Anzahl der Formeln das Hauptproblem)

Das Prinzip



3. Lösung zurückübersetzen

wieder möglicherweise automatisch

Beispiel 2: Sicherheitsprotokolle

Ziel: zwei Personen (Alice und Bob) wollen miteinander kommunizieren

- über ein **unsicheres** Daten- oder Telefonnetz,
- **sicher**, d. h., ohne daß ein Eindringling (Charlie) mithören oder sich als Alice oder Bob ausgeben kann.

Beispiel 2: Sicherheitsprotokolle

Ziel: zwei Personen (Alice und Bob) wollen miteinander kommunizieren

- über ein **unsicheres** Daten- oder Telefonnetz,
- **sicher**, d. h., ohne daß ein Eindringling (Charlie) mithören oder sich als Alice oder Bob ausgeben kann.

Hilfsmittel: Verschlüsselung

- Alice und Bob vereinbaren einen gemeinsamen Schlüssel und nutzen ihn, um ihr Gespräch zu verschlüsseln.
- Nur wer den Schlüssel kennt, kann das Gespräch entschlüsseln.

Beispiel 2: Sicherheitsprotokolle

Problem: wie kommen die Gesprächspartner an den gemeinsamen Schlüssel?

- Persönliche Übergabe kommt nicht immer in Frage.
- Wird der gemeinsame Schlüssel über das Netz unverschlüsselt verschickt, könnte Charlie ihn abfangen oder austauschen.
- Annahme: es gibt eine sichere Schlüsselzentrale, mit der Alice und Bob jeweils einen gemeinsamen Schlüssel vereinbart haben.

Beispiel 2: Sicherheitsprotokolle

Das folgende Schlüsselaustauschverfahren wurde 1993 von den beiden Kryptographen Neuman und Stubblebine vorgeschlagen:

Schritt 1:

Alice schickt (offen) Identifikation und Zufallszahl an Bob.

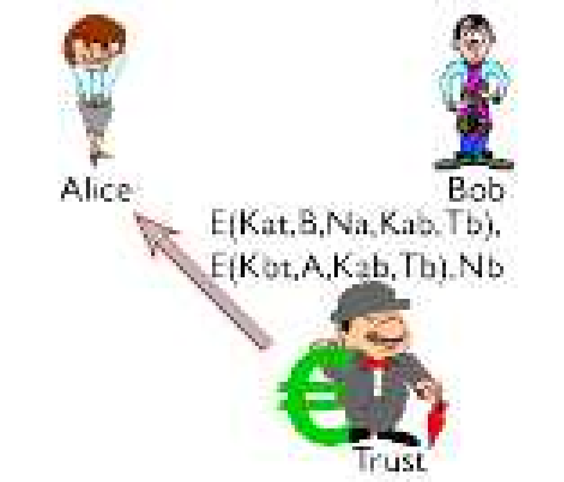
Beispiel 2: Sicherheitsprotokolle

Schritt 2:

Bob leitet Nachricht weiter an Schlüsselzentrale („Trust“).

Beispiel 2: Sicherheitsprotokolle

Schritt 3:



Trust schickt Nachricht an Alice. Darin: ein neuer gemeinsamer Schlüssel, einmal für Alice und einmal für Bob verschlüsselt.

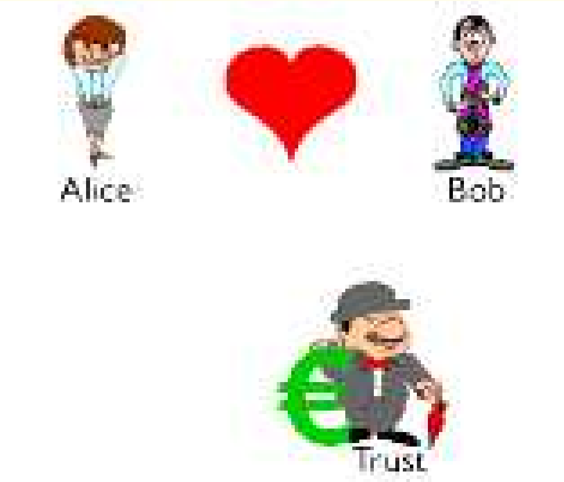
Beispiel 2: Sicherheitsprotokolle

Schritt 4:

Alice leitet den neuen gemeinsamen Schlüssel weiter an Bob.

Beispiel 2: Sicherheitsprotokolle

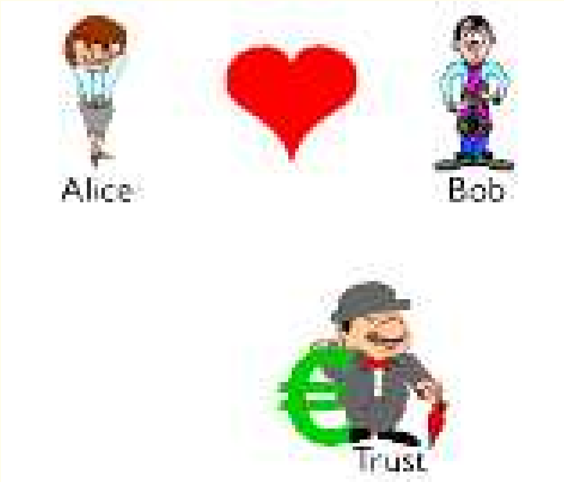
Schritt 5:



Alice und Bob können nun mit dem gemeinsamen Schlüssel kommunizieren.

Beispiel 2: Sicherheitsprotokolle

Ist das Verfahren sicher?



Wir übersetzen das Problem wieder in Formeln und lassen sie von einem Theorembeweiser untersuchen.

Beispiel 2: Sicherheitsprotokolle

Zuerst formalisieren wir die Eigenschaften des Protokolls:

- Wenn Alice/Bob/Trust eine Nachricht in einem bestimmten Format bekommt, dann schickt er/sie eine andere Nachricht ab.
- Wenn eine Nachricht übermittelt wird, kann Charlie sie mithören.
- Wenn Charlie eine verschlüsselte Nachricht bekommt und den passenden Schlüssel hat, kann er sie entschlüsseln.
- Wenn Charlie eine Nachricht hat, dann kann er sie an Alice/Bob/Trust abschicken.

...

Beispiel 2: Sicherheitsprotokolle

Dann formalisieren wir das Problem:

> Kann Charlie an einen Schlüssel kommen, den Bob für sicher hält?

Beispiel 2: Sicherheitsprotokolle

Dann formalisieren wir das Problem:

> Kann Charlie an einen Schlüssel kommen, den Bob für sicher hält?

Wir bekommen ungefähr 20 Formeln und geben sie in einen Theorembeweiser ein.

Beispiel 2: Sicherheitsprotokolle

Dann formalisieren wir das Problem:

> Kann Charlie an einen Schlüssel kommen, den Bob für sicher hält?

Wir bekommen ungefähr 20 Formeln und geben sie in einen Theorembeweiser ein.

Dieser antwortet nach 1 Sekunde „ja“!
Warum?

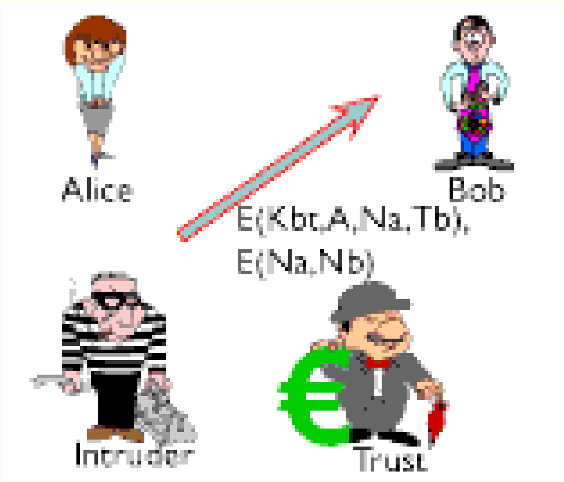
Beispiel 2: Sicherheitsprotokolle

Was kann passieren?

Charlie fängt die letzte Nachricht von Alice ab.

Beispiel 2: Sicherheitsprotokolle

Was kann passieren?



Charlie schickt eine veränderte Nachricht an Bob.

Beispiel 2: Sicherheitsprotokolle

Was kann passieren?

Bob entschlüsselt die Nachricht und denkt, sie komme von Alice.

Beispiel 2: Sicherheitsprotokolle

Was kann passieren?

Bob startet Kommunikation mit Alice ...

Beispiel 2: Sicherheitsprotokolle

Was kann passieren?

... aber spricht in Wirklichkeit mit Charlie.

Beispiel 2: Sicherheitsprotokolle

Der Fehler ist tatsächlich leicht zu beheben.

Mit ein wenig Erfahrung erkennt man aus der Ausgabe des Theorembeweisers, wie es geht.

Modellüberprüfung

Bisher sahen die Probleme so aus:

Folgt eine Formel (z. B.: $\mathsf{ermordet}(a, a)$) aus anderen Formeln?

Daneben ist aber auch die folgende Fragestellung interessant:

Hat eine vollständige Beschreibung („Modell“) einer Software oder Hardware eine (durch eine Formel beschriebene) Eigenschaft?

Beispiel 3: Handy

Ein Handy kann sich in verschiedenen Zuständen befinden.
Beispielsweise:

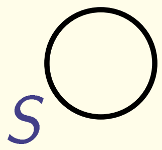
- Startzustand (direkt nach dem Einschalten).
- Benutzer gibt Telefonnummer ein.
- Telefonnummer ist vollständig eingegeben;
  Handy versucht, Basisstation zu erreichen.
- Kontakt zur Basisstation aufgenommen;
  anderer Teilnehmer wird angerufen.
- Gespräch wird geführt.

  ...

Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

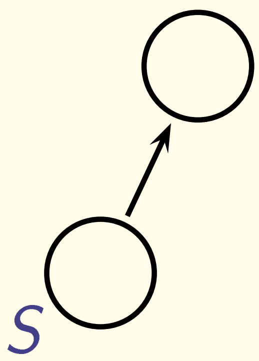
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

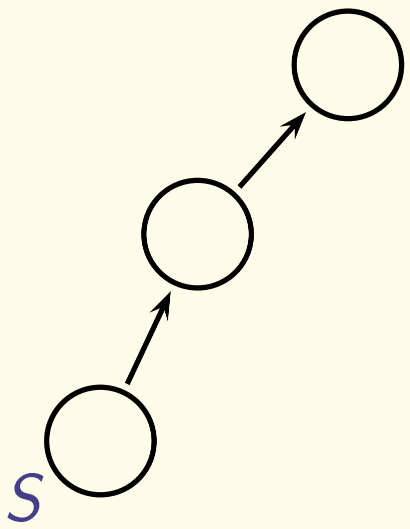
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

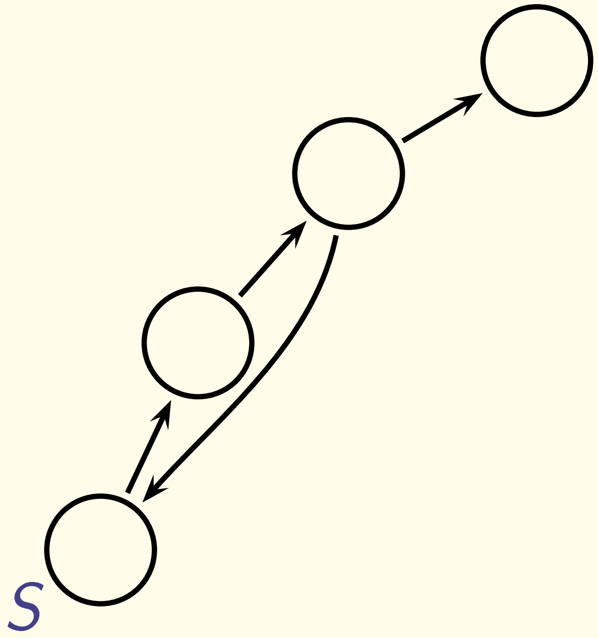
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

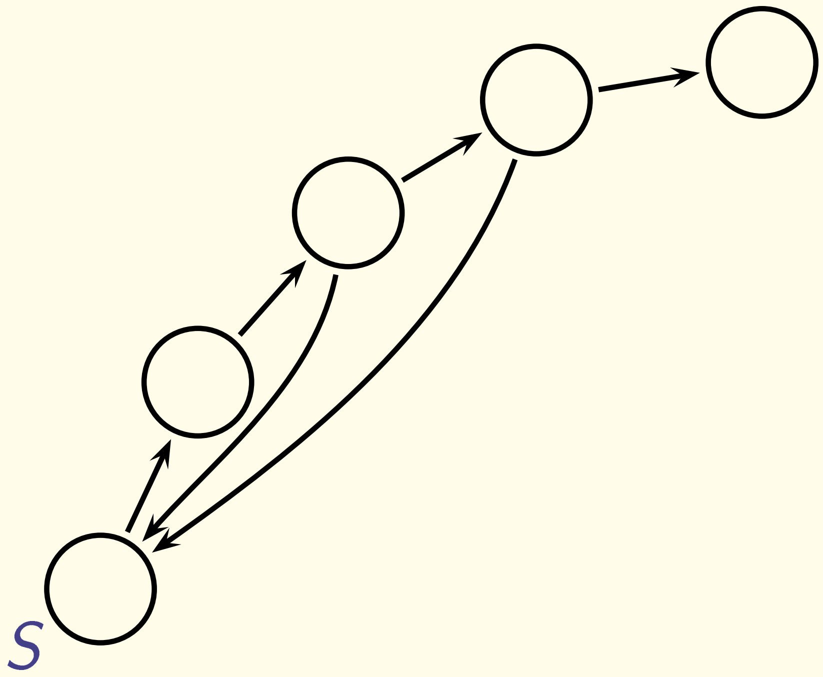
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

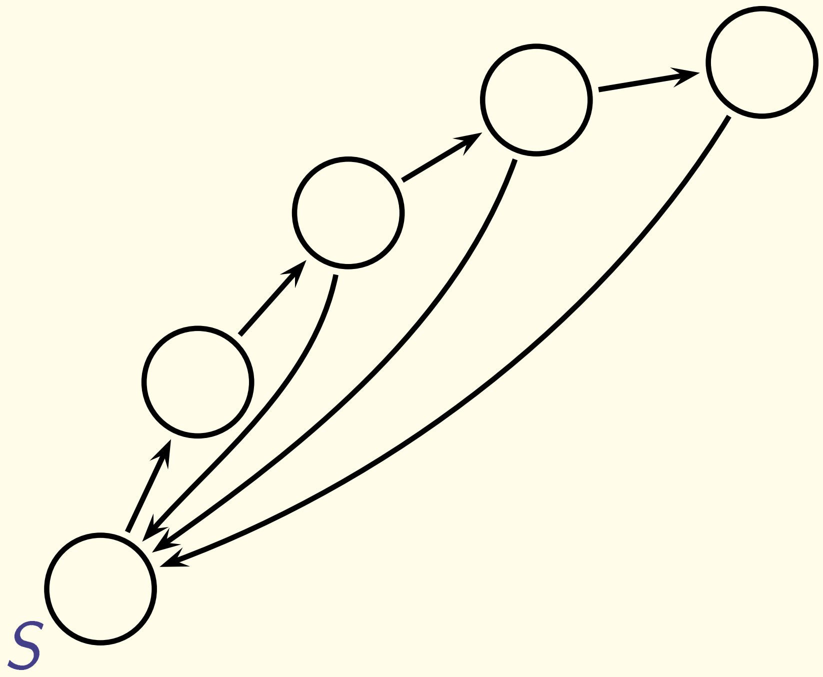
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

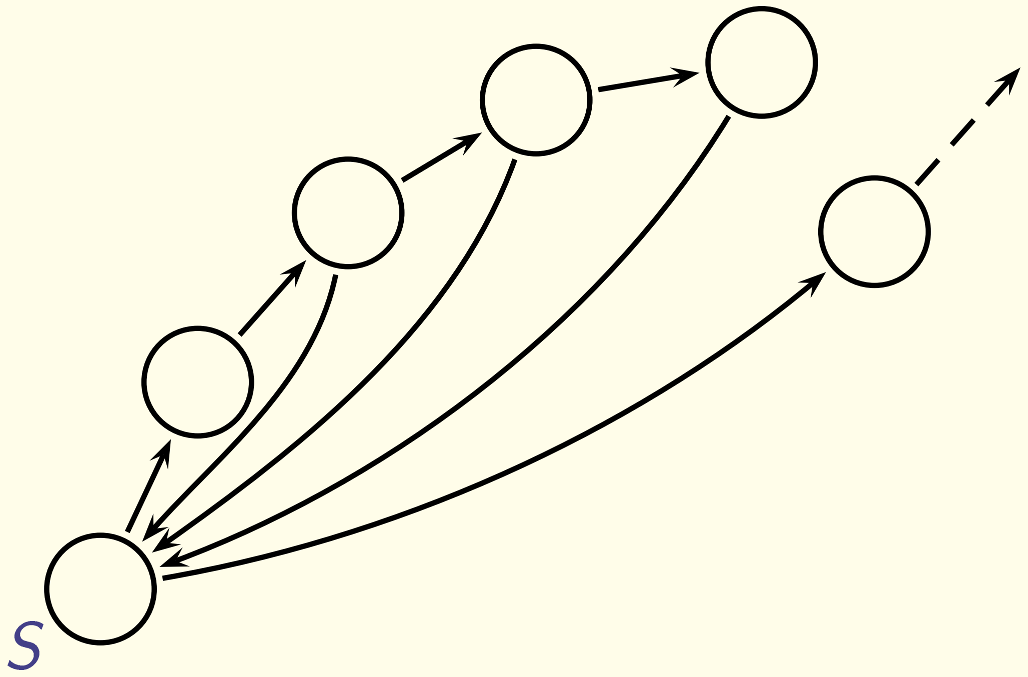
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

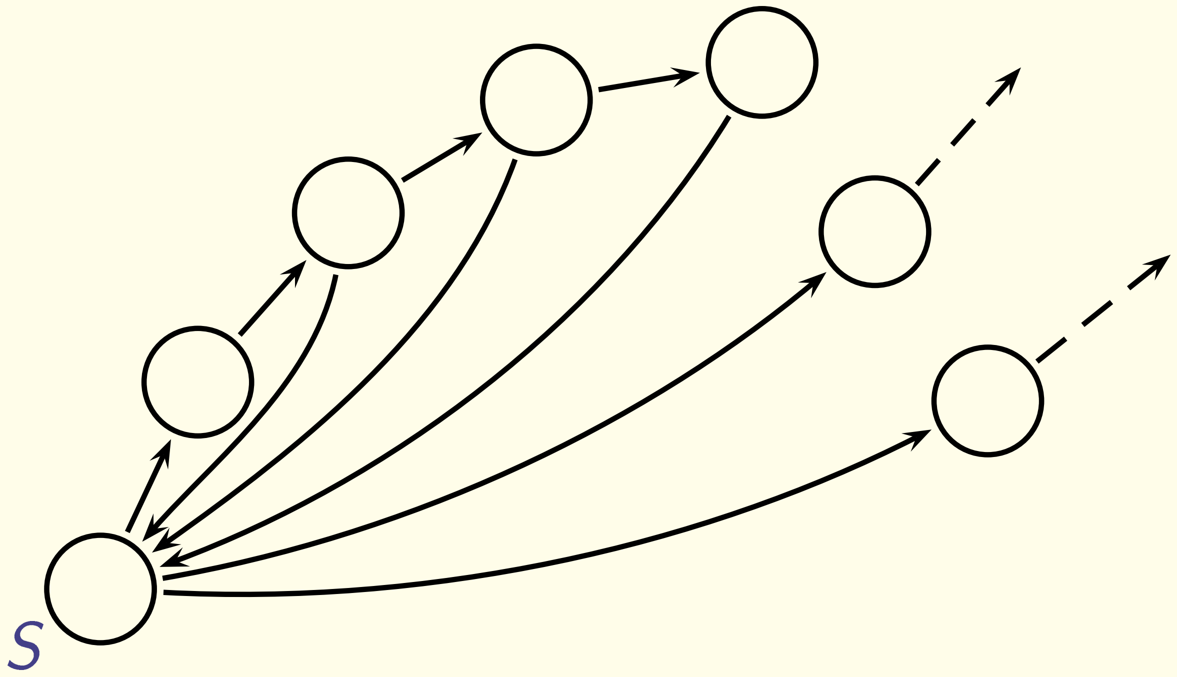
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

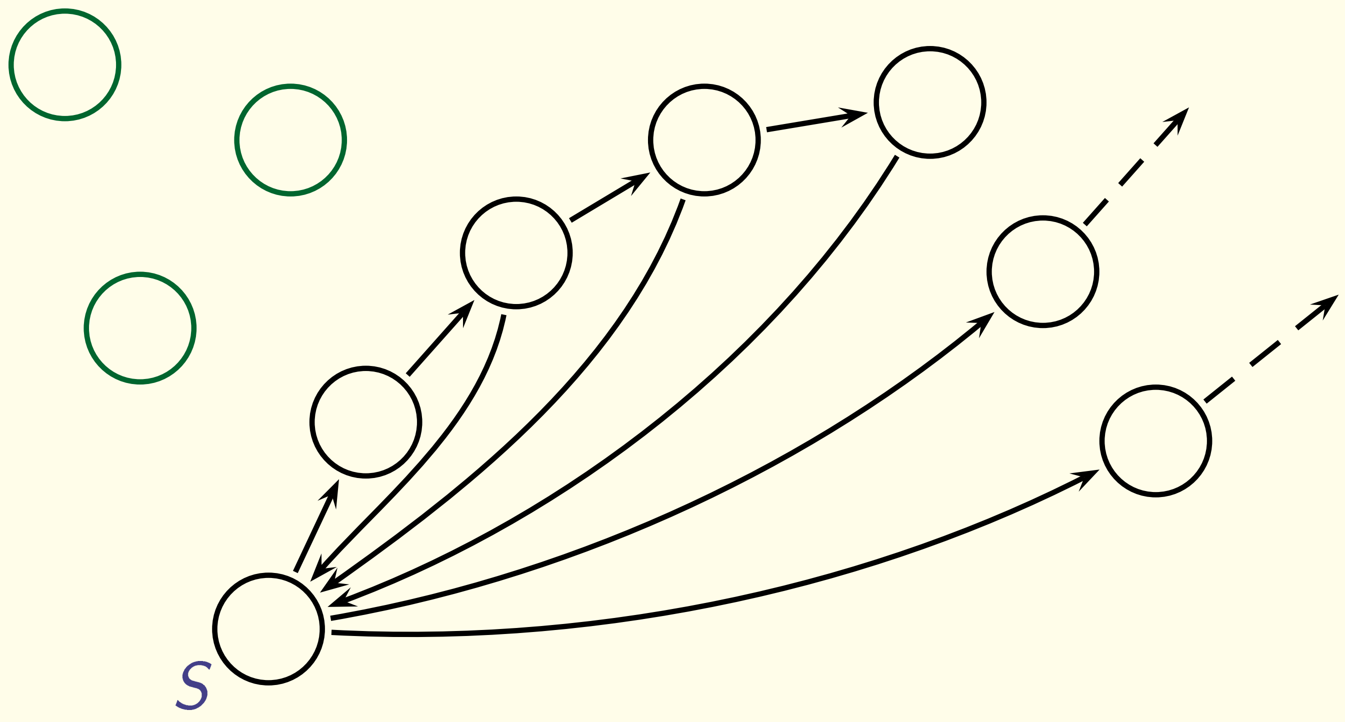
Wir stellen die Zustände durch Kreise dar.

Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



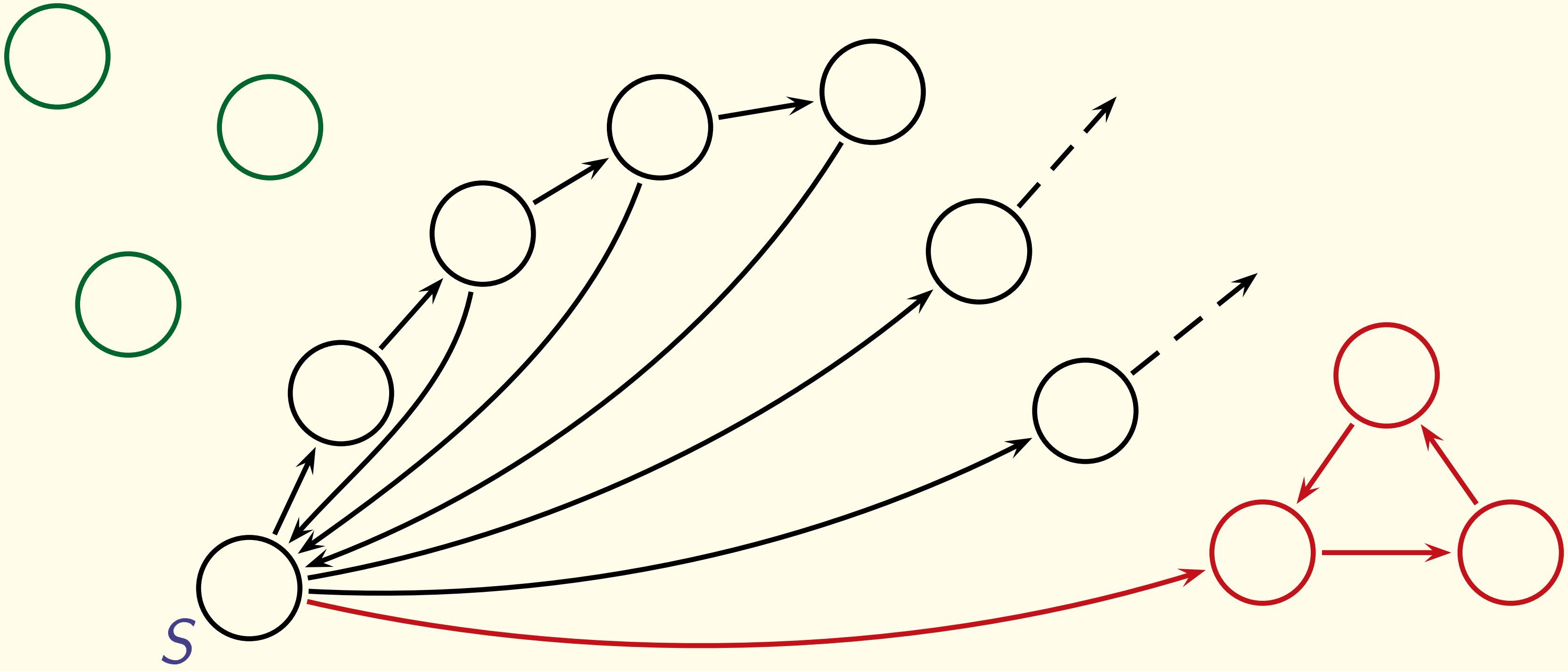
Beispiel 3: Handy

Es kann auch „unmögliche“ Zustände geben.
Aber diese dürfen dann natürlich nicht vom Startzustand aus erreichbar sein.



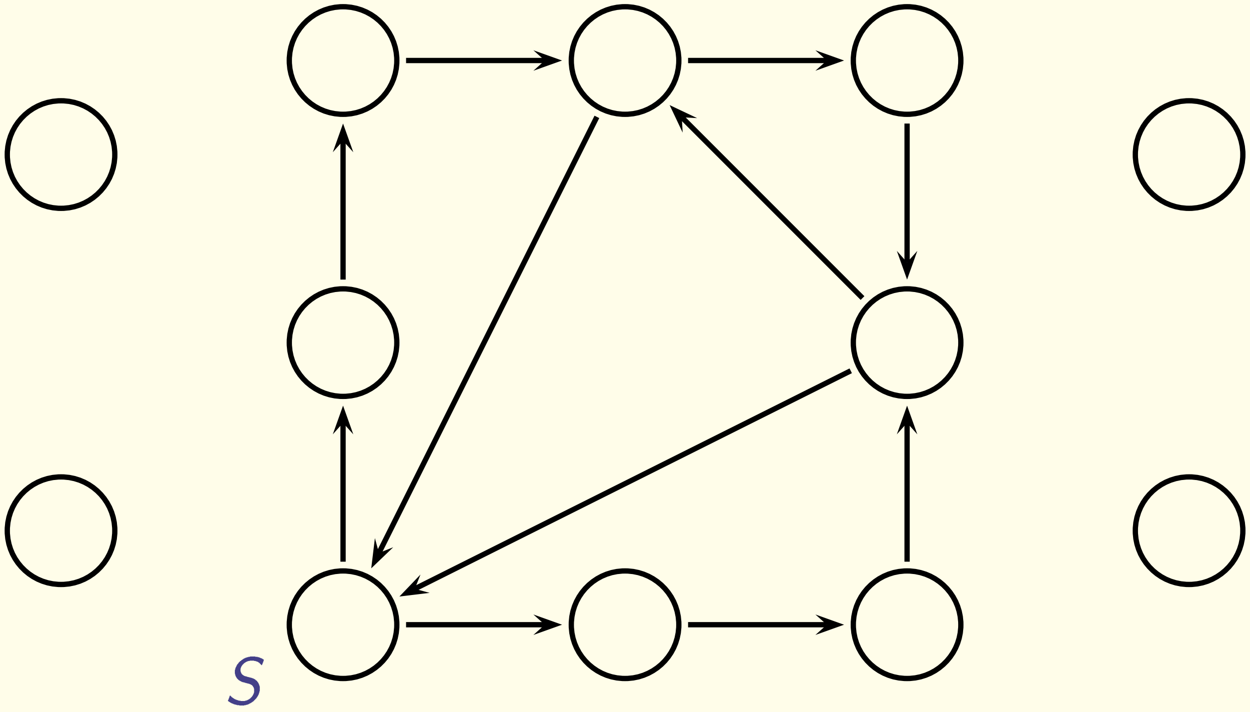
Beispiel 3: Handy

Was es aber auf keinen Fall geben darf, das sind Sackgassen.



Beispiel 3: Handy

Frage: Wie kann man automatisch feststellen, ob irgendein Zustandsdiagramm eine Sackgasse besitzt?

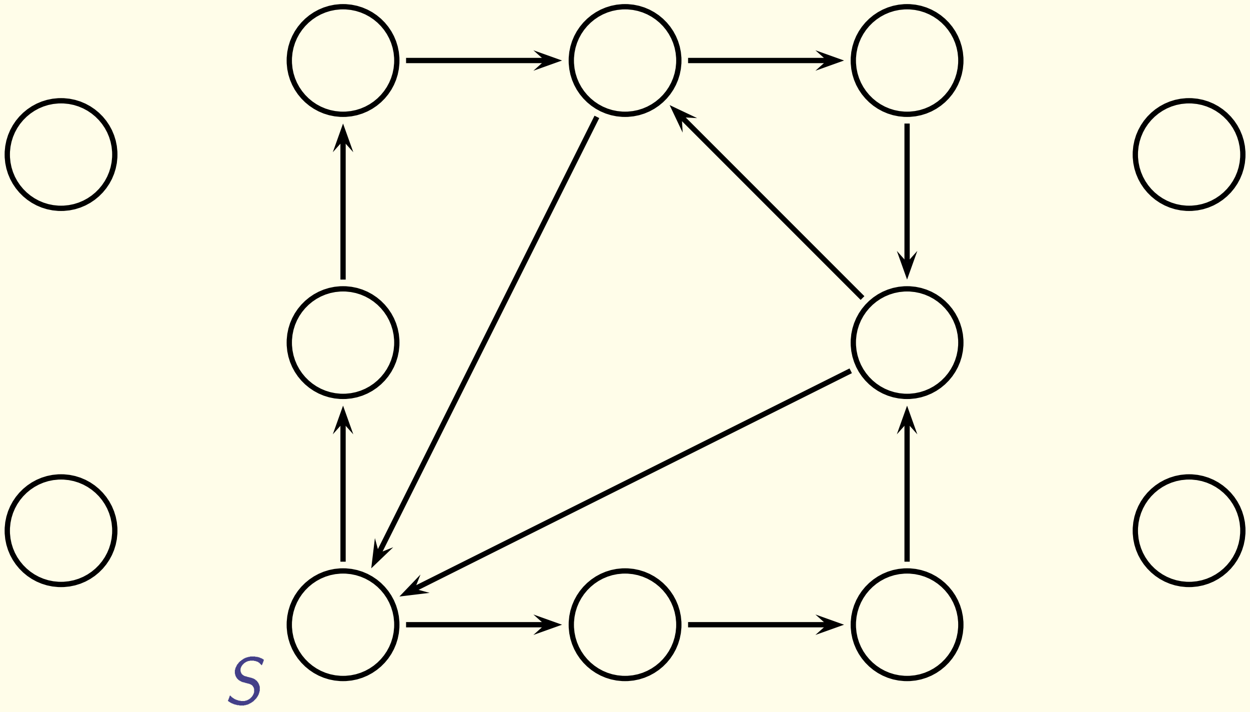


Beispiel 3: Handy

Als Formel ausgedrückt: Hat S die Eigenschaft $\mathsf{EF} \neg \mathsf{EF}\, S$?

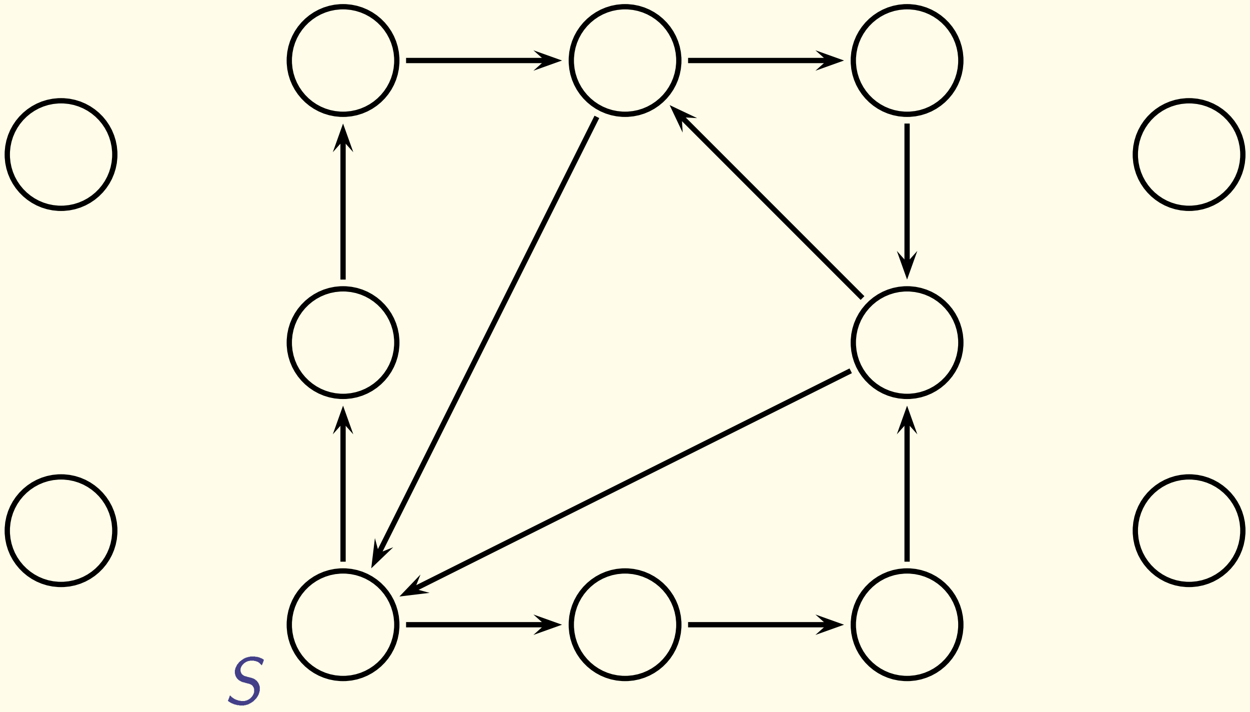
Dabei bedeutet EF:

„man kann einen Zustand erreichen mit der Eigenschaft . . . “



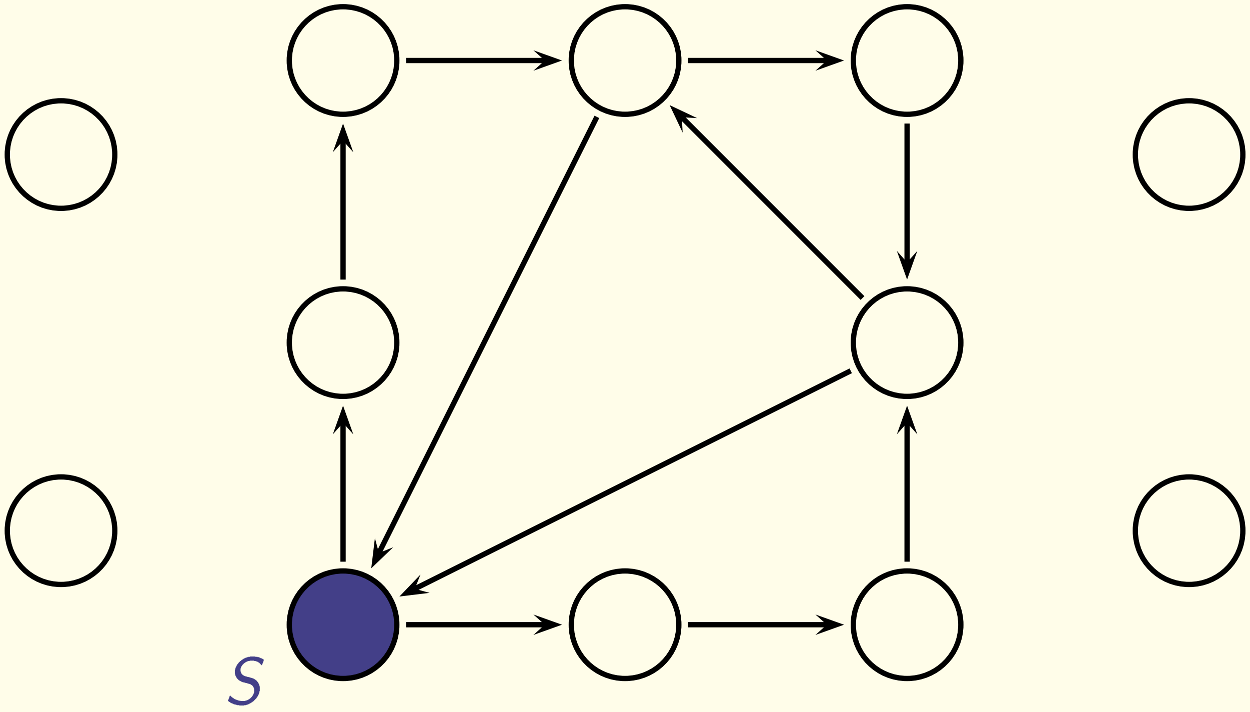
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S, das heißt, alle Zustände, von denen aus man S erreichen kann.



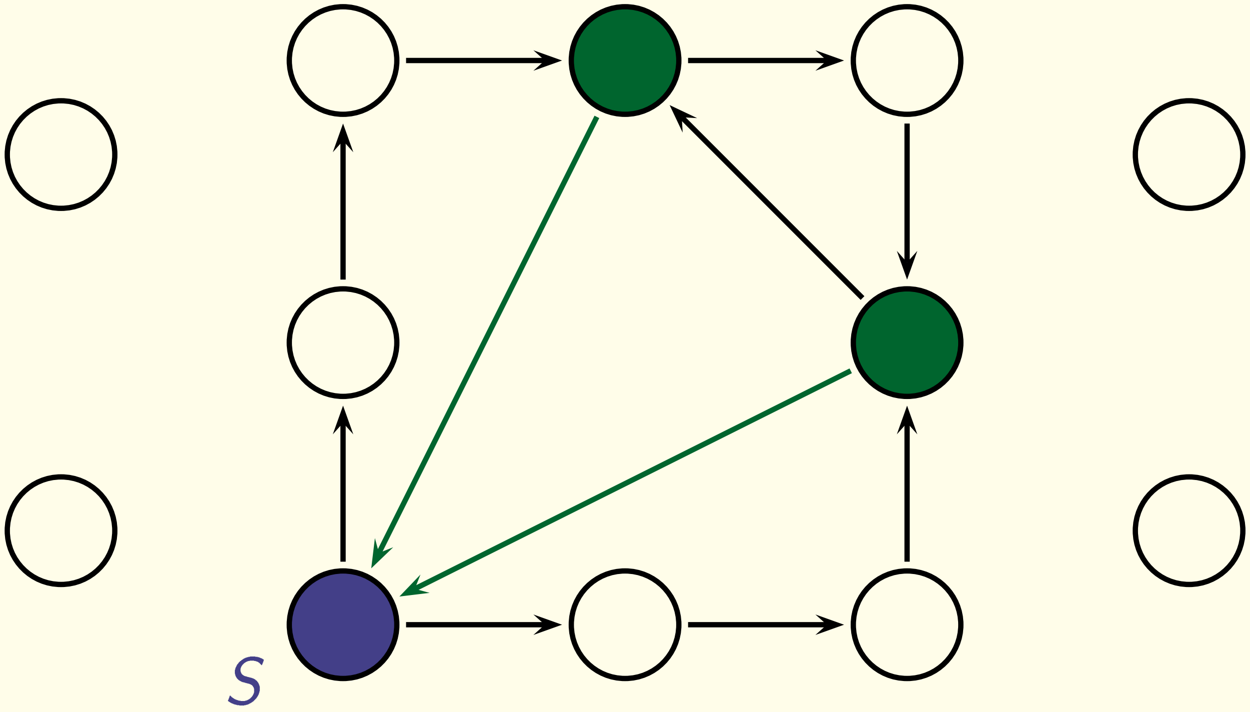
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S, das heißt, alle Zustände, von denen aus man S erreichen kann.



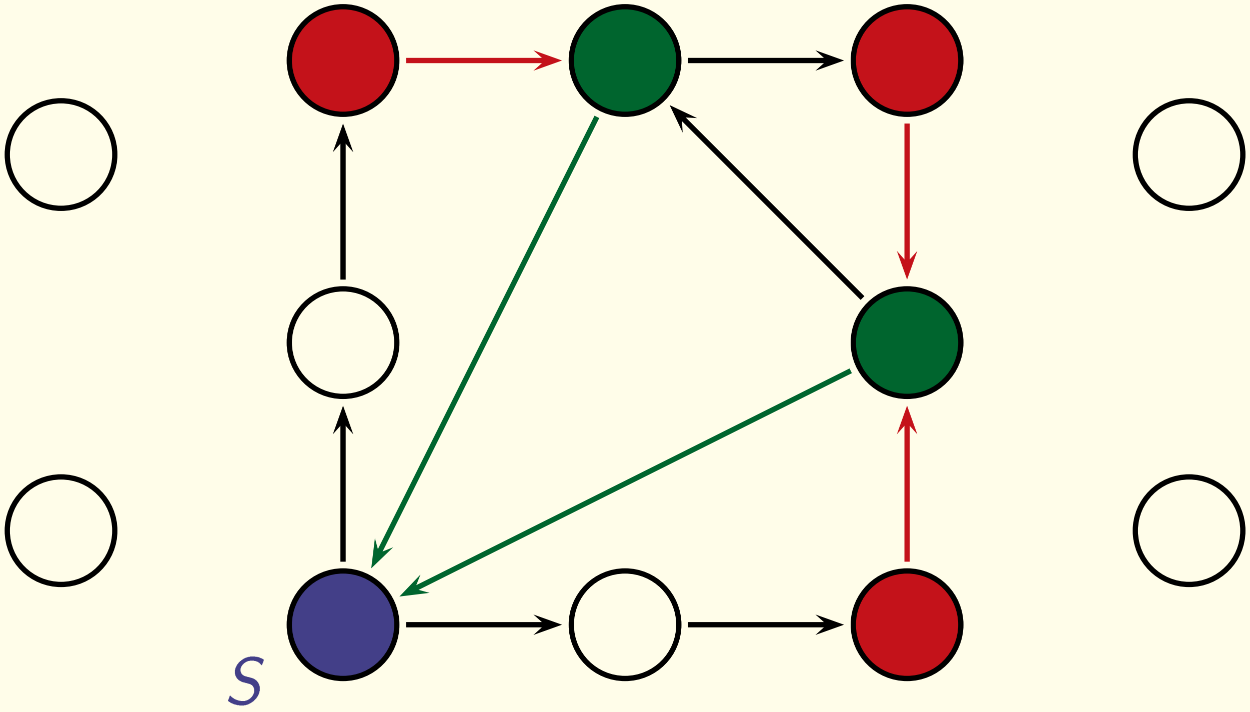
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S, das heißt, alle Zustände, von denen aus man S erreichen kann.



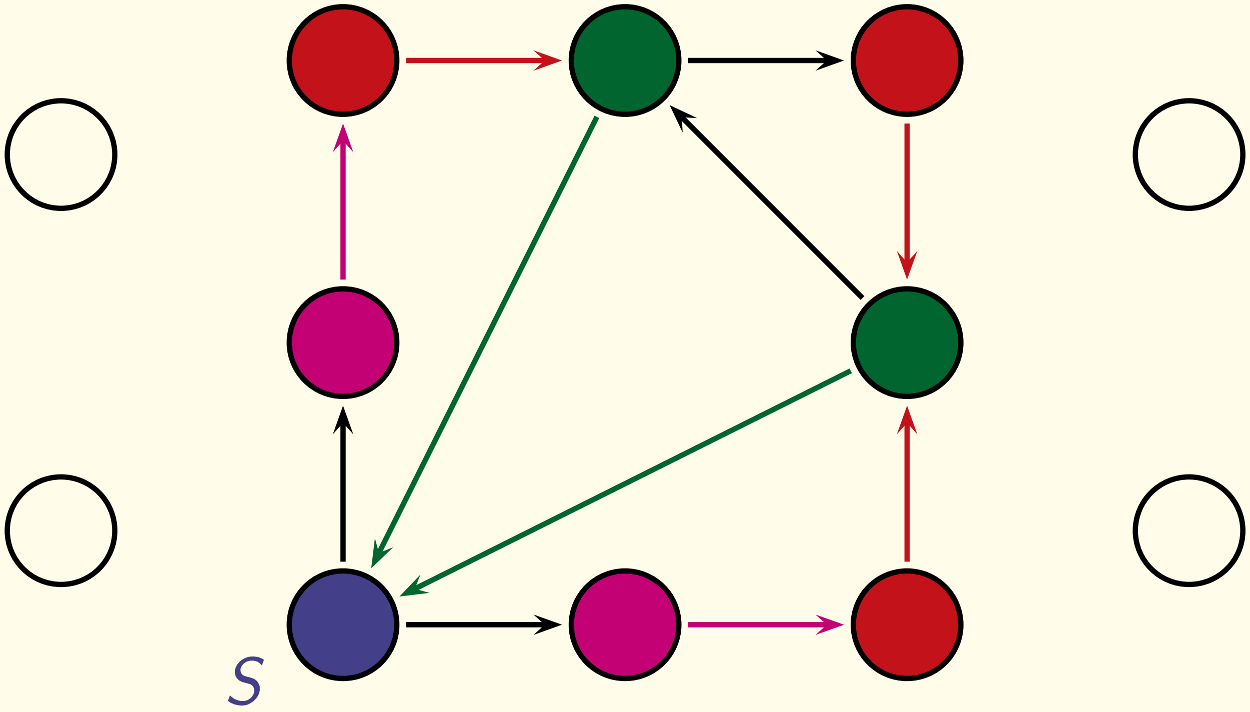
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S, das heißt, alle Zustände, von denen aus man S erreichen kann.



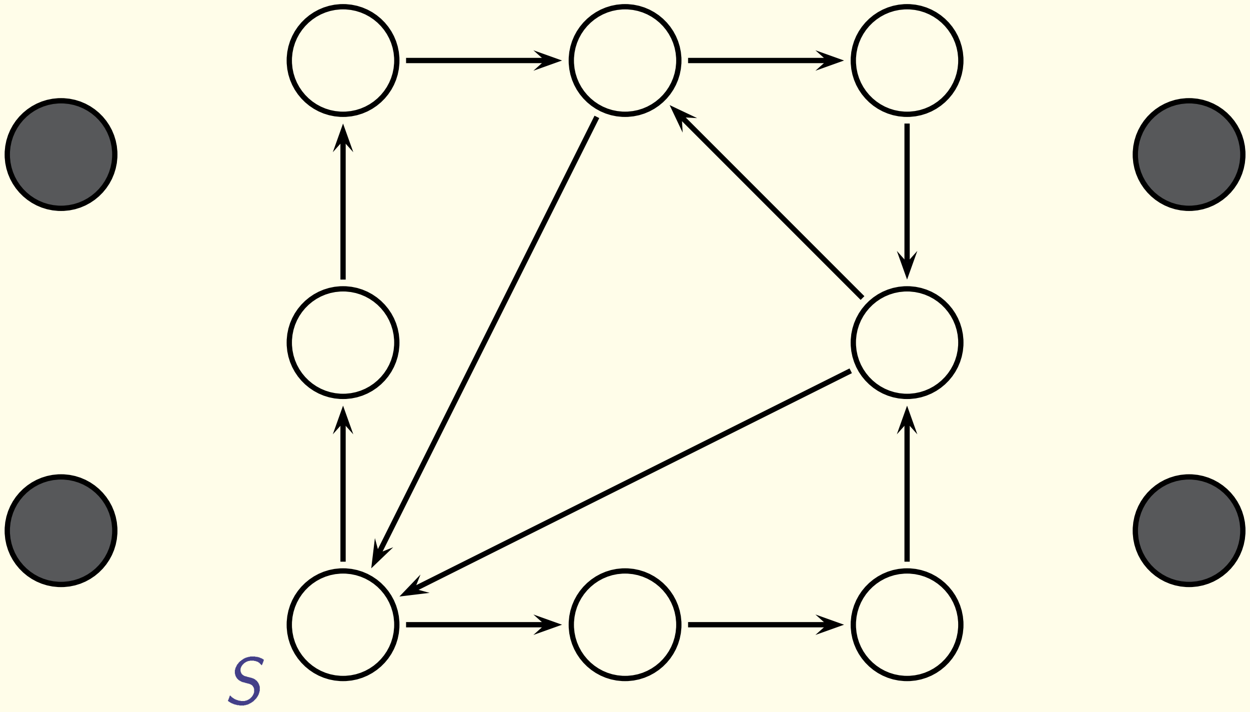
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S, das heißt, alle Zustände, von denen aus man S erreichen kann.



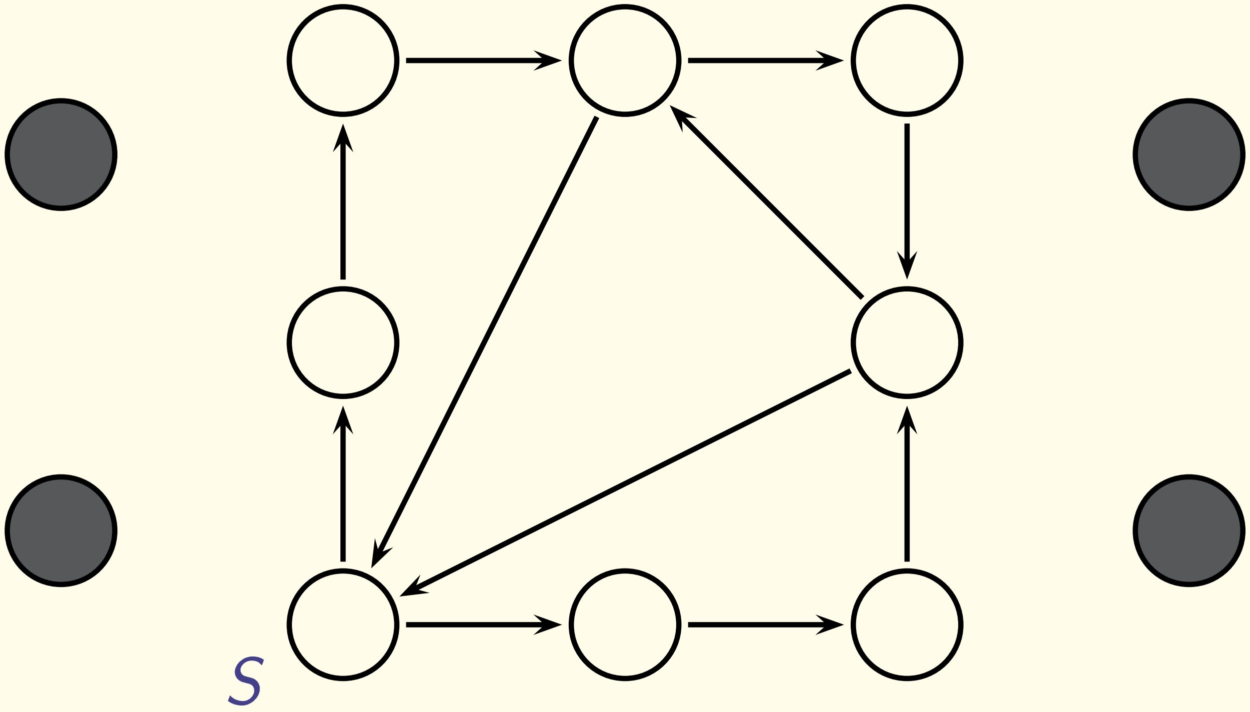
Beispiel 3: Handy

Dann ermitteln wir alle Zustände mit der Eigenschaft $\neg$ EF S, das heißt, alle Zustände, die **nicht** die Eigenschaft EF S haben.



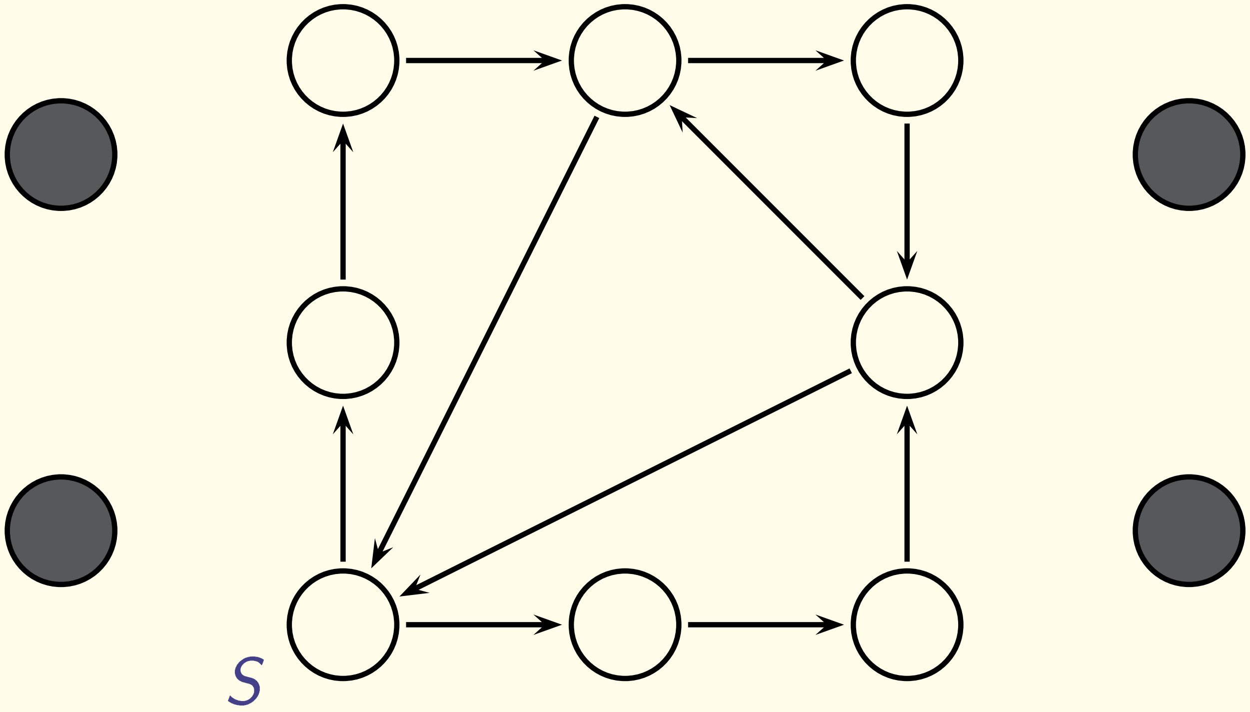
Beispiel 3: Handy

Zum Schluß ermitteln wir alle Zustände mit der Eigenschaft $\mathsf{EF} \neg \mathsf{EF}\, S$, also diejenigen, von denen aus man einen grauen Zustand erreicht. (Das sind aber gerade die grauen Zustände selbst.)



Beispiel 3: Handy

Da der Zustand S **nicht** zu diesen Zuständen gehört, wissen wir nun, daß wir von S aus keine Sackgasse erreichen.



Modellüberprüfung

Modellüberprüfung ist heute für jeden Chipentwickler eine Standardmethode.

Die Hauptschwierigkeit ist dabei die Größe der Probleme:

$10 \text{ bits} \mathrel{\hat{=}} 1\,024 \text{ Zustände}$

$20 \text{ bits} \mathrel{\hat{=}} 1\,048\,576 \text{ Zustände}$

$30 \text{ bits} \mathrel{\hat{=}} 1\,073\,741\,824 \text{ Zustände}$

$40 \text{ bits} \mathrel{\hat{=}} 1\,099\,511\,627\,776 \text{ Zustände}$

Modellüberprüfung

Modellüberprüfung ist heute für jeden Chipentwickler eine Standardmethode.

Die Hauptschwierigkeit ist dabei die Größe der Probleme:

10 bits $\widehat{=}$ 1 024 Zustände
20 bits $\widehat{=}$ 1 048 576 Zustände
30 bits $\widehat{=}$ 1 073 741 824 Zustände
40 bits $\widehat{=}$ 1 099 511 627 776 Zustände

Ausweg:

Zustände und Übergange nicht explizit darstellen, sondern Zustandsmengen durch (hoffentlich kleine) Formeln kodieren.

Modellüberprüfung

Noch kniffliger:

- Software-Verifikation
- Hybride Systeme (d. h., Systeme, in denen Zustände auch durch reelle Zahlen beschrieben werden)

In beiden Fällen ist die Anzahl der Zustände unendlich.

Modellüberprüfung

Noch kniffliger:

- Software-Verifikation
- Hybride Systeme (d. h., Systeme, in denen Zustände auch durch reelle Zahlen beschrieben werden)

In beiden Fällen ist die Anzahl der Zustände unendlich.

Ausweg:

Zustände zu endlich vielen Intervallen oder Klassen zusammenfassen (funktioniert aber leider nicht immer).

Logik in der Informatik

Was ist Logik?

- Mathematisch?

  ja

- Unverständlich?

  hoffentlich nein

- Reine Theorie ohne praktischen Nutzen?

  nein: Verifikation von Hardware, Software, Protokollen;
  Sprachverarbeitung und Wissensrepräsentation;
  Abfragesprachen für Datenbanken; ...