Theoretische Grundlagen des Software Engineering

0: Organisatorisches & Einführung

Stephan Schulz
schulz@eprover.org

Software Systems Engineering

Stiftung Universität Hildesheim 2003

Kurzvorstellung

Stephan Schulz

- Studium der Informatik in Kaiserslautern
- Promotion an der TU München
- Forschung: Automatische Deduktion
- Lehre: U. Miami, U. West Indies
- Nebenberuf: Projektleiter COMSOFT
  - Entwicklungsprojekte Flugsicherung

Veranstaltungstermine

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
10.15-11.45	TGSE VL (C-213)				
12.00-12.45	TGSE Übung (C-213)				
14.15-15.45	TGSE VL (C-213)				
16.00-16.45	TGSE Übung (C-213)				

Oktober	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
November	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Dezember	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Januar	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Kurzfristige Änderungen möglich, aber nicht erwartet

Unterlagen

Vorlesungsskript und Übungen:

- Hoffentlich bald per LernWeb/Moodle
- Vorläufig unter

  http://www.eprover.org/TEACHING/TGSE2009.html

Übungen

Initial Präsenzübungen

- Aufgaben werden in der Stunde gestellt, bearbeitet, und diskutiert
- Regelmäßige aktive Teilnahme ist Voraussetzung für die Zulassung zur Prüfung!

Leistungsbewertung

Je nach Anzahl der Teilnehmer Klausur oder mündliche Prüfung

- Ca. eine Woche nach Abschluss der Vorlesung
- Geplanter Termin: 1. Februar 2010

Bewertungskriterien:

- Reproduktion (30%)
- Verständnis (40%)
- Transfer und Anwendung (30%)

Ziele der Vorlesung

- Vokabular der Domäne bereitstellen
  - Unverzichtbare Voraussetzung
- Methoden
  - Wesentliches Ziel
- Ergebnisse
  - Nett zu wissen, besser herzuleiten!

- Erschließt die Literatur
- Effizienterer Gedankenaustausch

Vokabular

- Höheres Abstraktionsniveau
- “Weiterdenken”

Image credit: Stanislav Doronenko via the Wikimedia Commons

Definition

Eine Definition ist eine genaue Beschreibung eines Objektes oder Konzepts.

- Definitionen können einfach oder komplex sein
- Definitionen müssen präzise sein - es muss klar sein, welche Objekte oder Konzepte beschrieben werden
- Oft steckt hinter einer Definition eine Intuition - die Definition versucht, ein "reales" Konzept formal zu beschreiben
  - Hilfreich für das Verständnis - aber gefährlich! Nur die Definition an sich zählt für formale Argumente

Methoden

Wie beweist man...

- dass alle C-Programme gleich oft (und) enthalten?
- dass es keinen regulären Ausdruck für C-Programme gibt?

Wie findet man...

- alle Treffer für einen regulären Ausdruck?
- alle Modelle einer aussagenlogischen Formel?

Wie beschreibt man...

- eine domänenspezifische Sprache?
- eine Vorbedingung für eine Prozedur?

Mathematische Beweise

Ein Beweis ist ein Argument, das einen *verständigen* und *unvoreingenommenen* Empfänger von der *unbestreitbaren Wahrheit* einer Aussage überzeugt.

- Oft mindestens semi-formell
- Aussage ist fast immer ein Konditional
  - ...aber die Annahmen sind für semi-formelle Beweise oft implizit

Ergebnisse

Kein endlicher Automat kann die Menge aller C-Programme erkennen

Die Menge der kontextfreien Sprachen ist eine echte Obermenge der rechtslinearen Sprachen

Aussagenlogik ist entscheidbar

Lokale Konfluenz und Termination implizieren Konvergenz

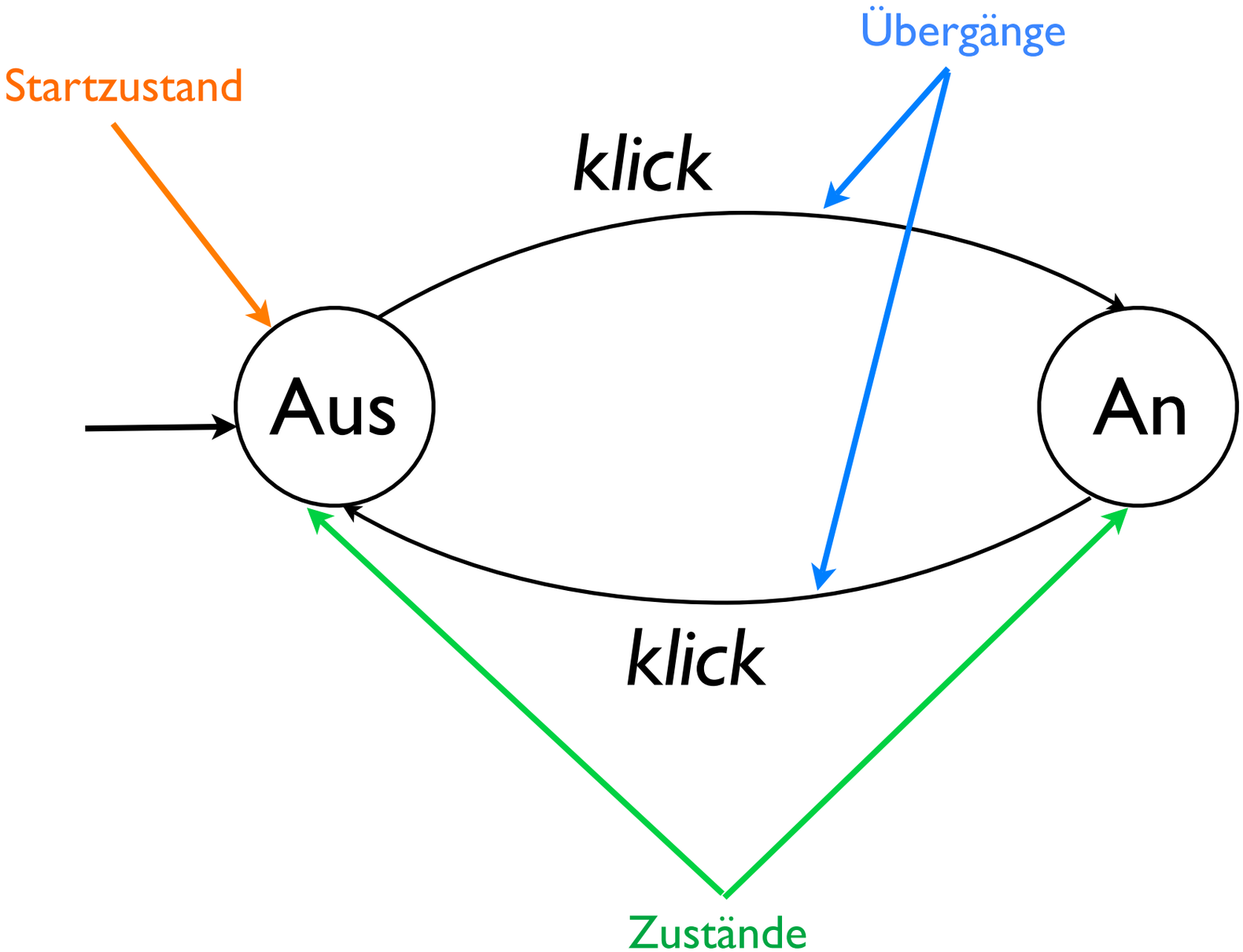
Ihre Erwartungen?

Inhalte

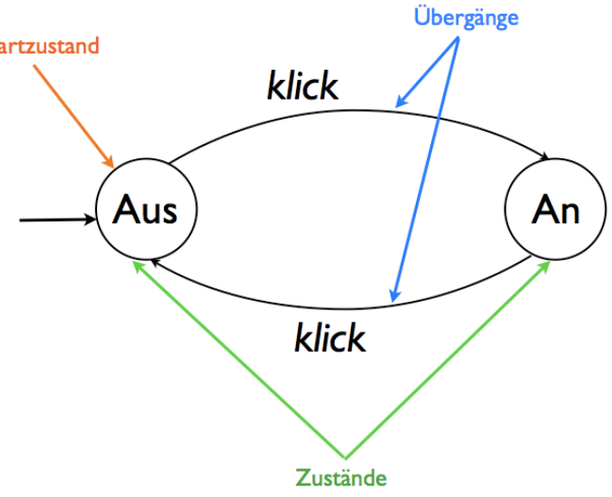
- Formale Sprachen, Automaten und Grammatiken
- Aussagenlogik und Prädikatenlogik
- Reduktionssysteme/Rewriting

Einführung Sprachen+Automaten

Endliche Automaten: Einfaches Beispiel



Endliche Automaten: Einfaches Beispiel



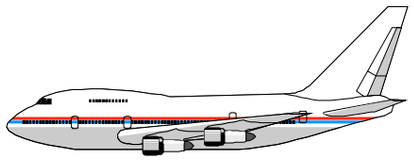
Formaler:

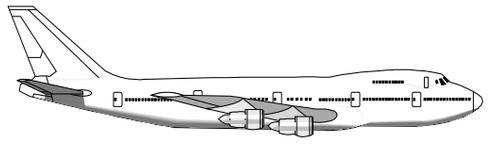
- Q={Aus, Ein} ist die Menge der Zustände
- Σ={klick} ist das Alphabet
- Übergangsfunktion

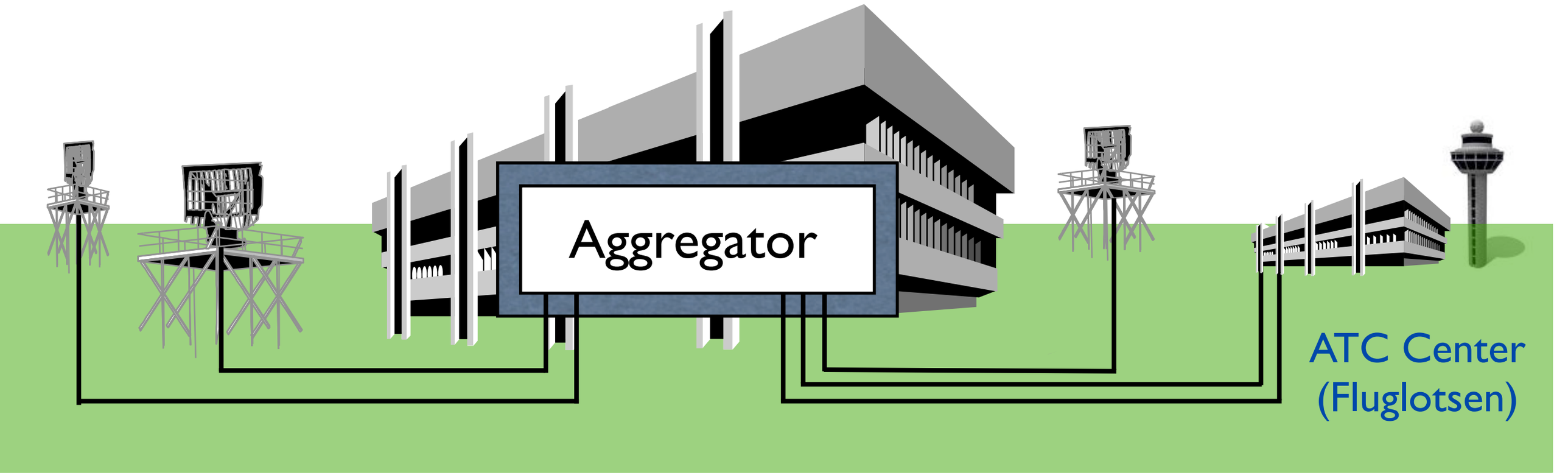
δ	klick
Aus	An
An	Aus

- Aus ist der Startzustand
- Es gibt keine akzeptierenden Zustände

Beispiel: Redundanzmanagement

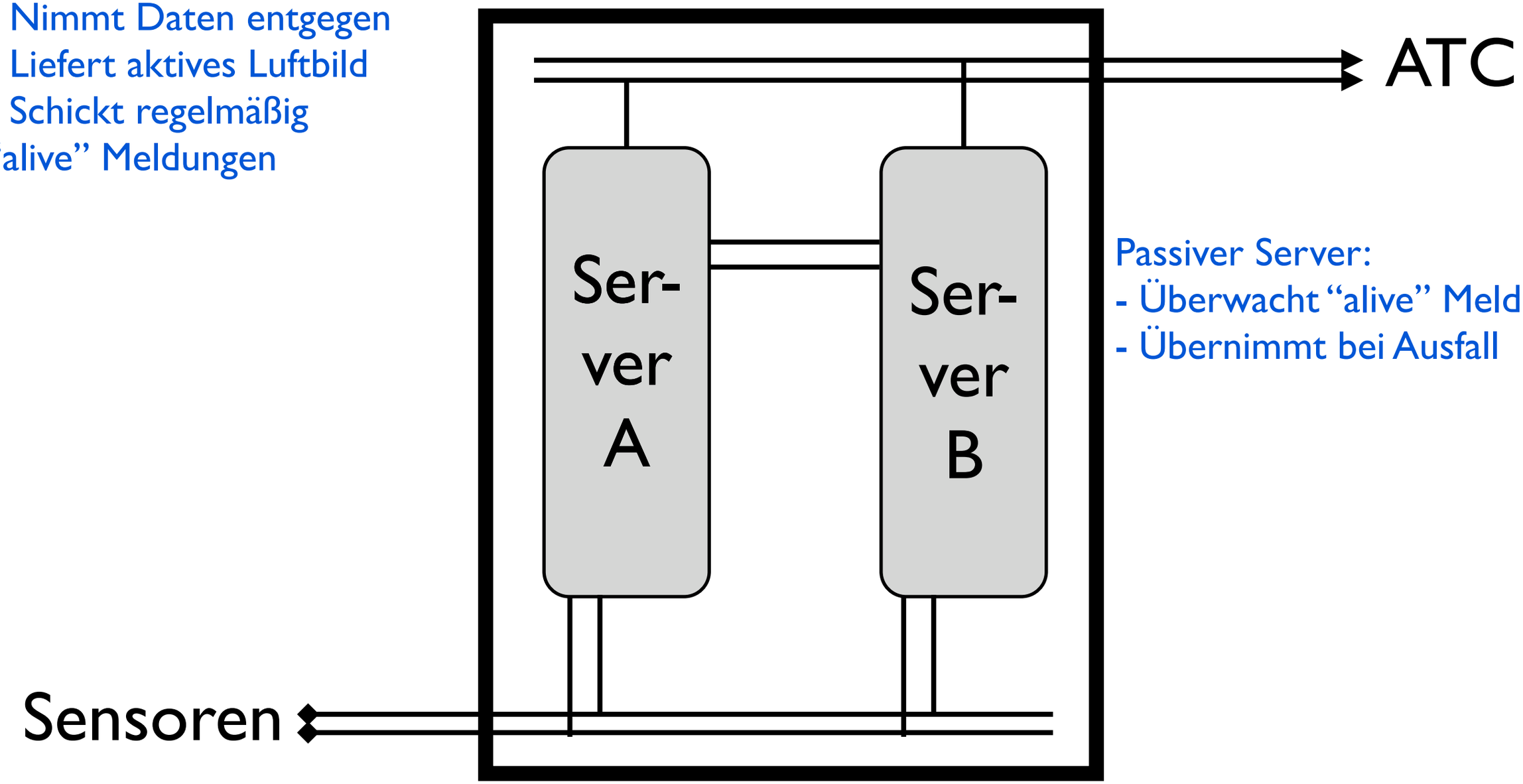






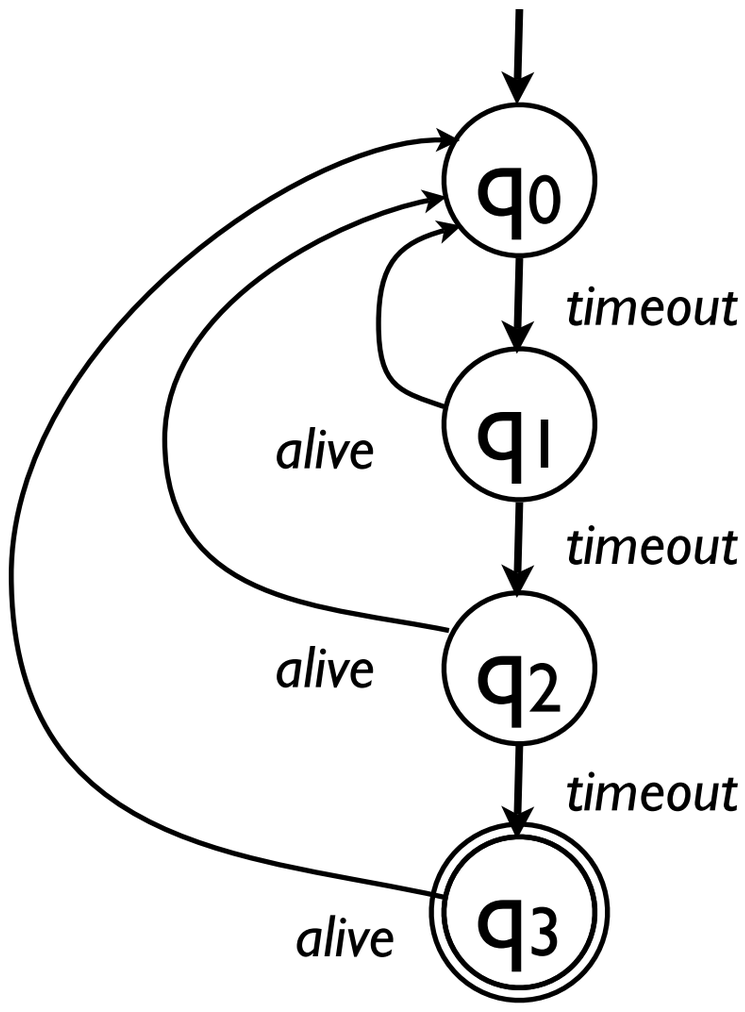
Beispiel: Redundanzmanagement

Aktiver Server:
- Nimmt Daten entgegen
- Liefert aktives Luftbild
- Schickt regelmäßig "alive" Meldungen



Passiver Server:
- Überwacht "alive" Meldungen
- Übernimmt bei Ausfall

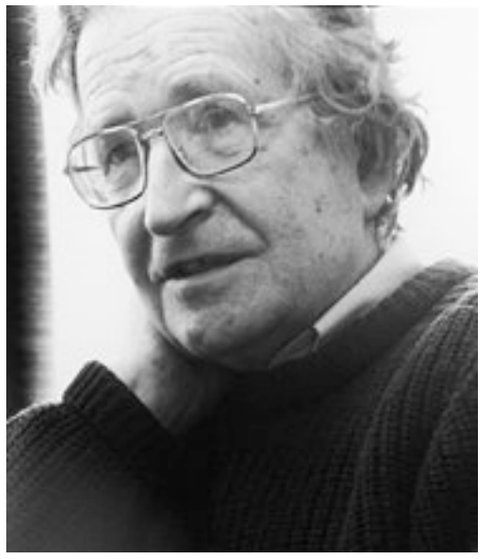
Zustandsdiagramm



Zwei Eingaben (“Buchstaben”)

- *timeout*: 0.1 Sekunden sind vergangen
- *alive*: Andere Server ist aktiv
- q_0, q_1, q_2: Server ist passiv
  - Keine Verarbeitung, keine alives
- Wenn q_3 erreicht wird:
  - Übername als aktiver Server (schicke *alives*)

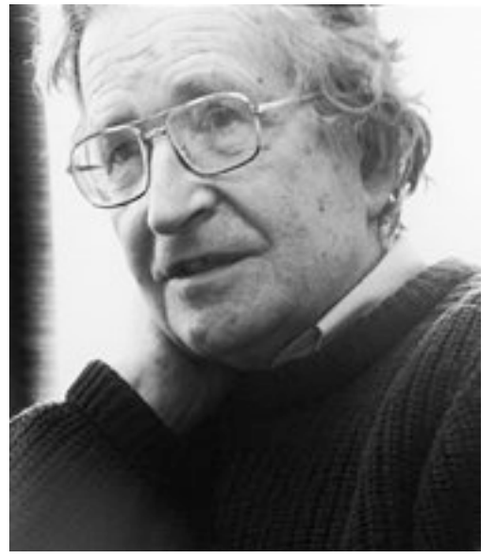
Formale Sprachen



Mengen von Worten über definierten Alphabeten

- Alle Namen im Telefonbuch von Hamburg
- Fließkommazahlen in wissenschaftlicher Notation: 1.3E+17, 1.0, 2e-13...
- Korrekt geklammerte Arithmetische Ausdrücke über *, +, x,y,z: 1+2, 1*(2+3), ((2+2)+2)+3...
- Worte der Form a...a, wobei die Anzahl der Wiederholungen von a eine Primzahl ist
- Alle (C-)Programme, die jemals terminieren
- Alle C-Programme, die niemals terminieren

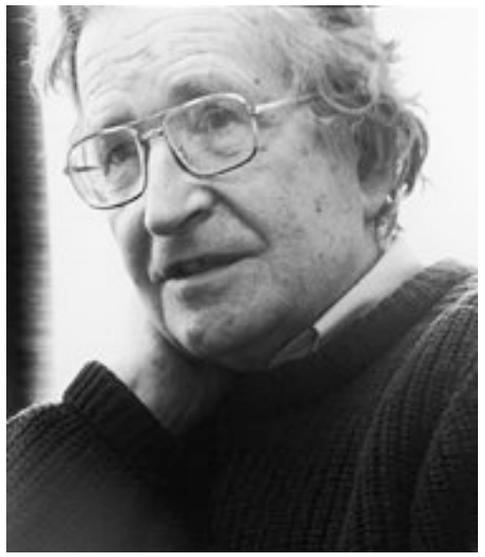
Formale Sprachen



Beschreibung von Sprachen:

- Generierend
  - Formale Grammatiken
    - Praktische Ausprägungen: BNF, EBNF
- Akzeptierend:
  - Automaten
    - Praktische Ausprägungen: Lexer, Parser
  - Reguläre Ausdrücke

Noam Chomsky (*1928)



Idee: "Universelle Grammatik" (1950er)

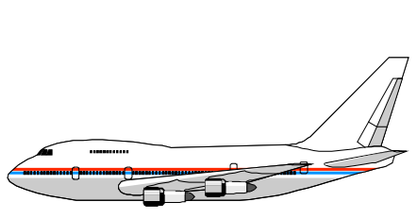
- Vermutung: Beschreibt Gemeinsamkeiten aller menschlichen Sprachen
  - Bis heute diskutiert

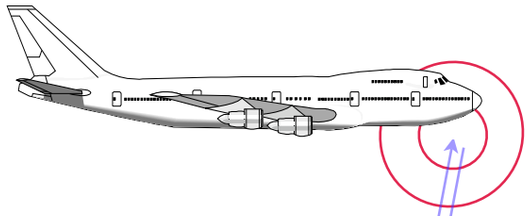
Aber: Massiven Einfluss im Bereich der Informatik

- Nahezu jede Sprache von XML über SQL und Scheme bis C und JAVA wird mit formalen Grammatiken beschrieben

Einführung Logik

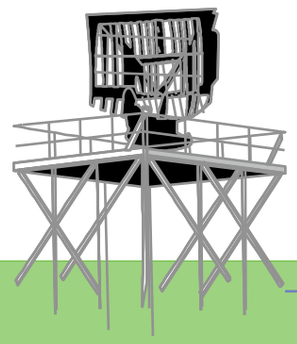
Beispiel Flugsicherung





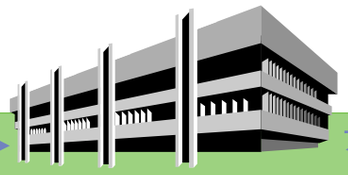
Kontrollierter Luftraum

Unkontrollierter Luftraum



Überwachungssystem

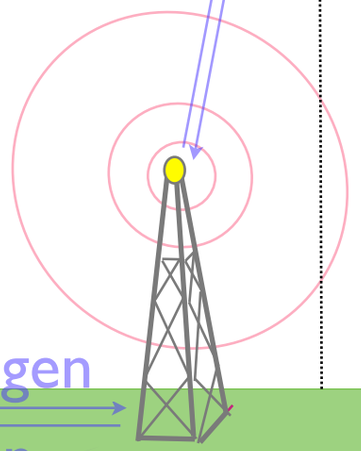
Luftlagebild



ATC Center (Fluglotsen)

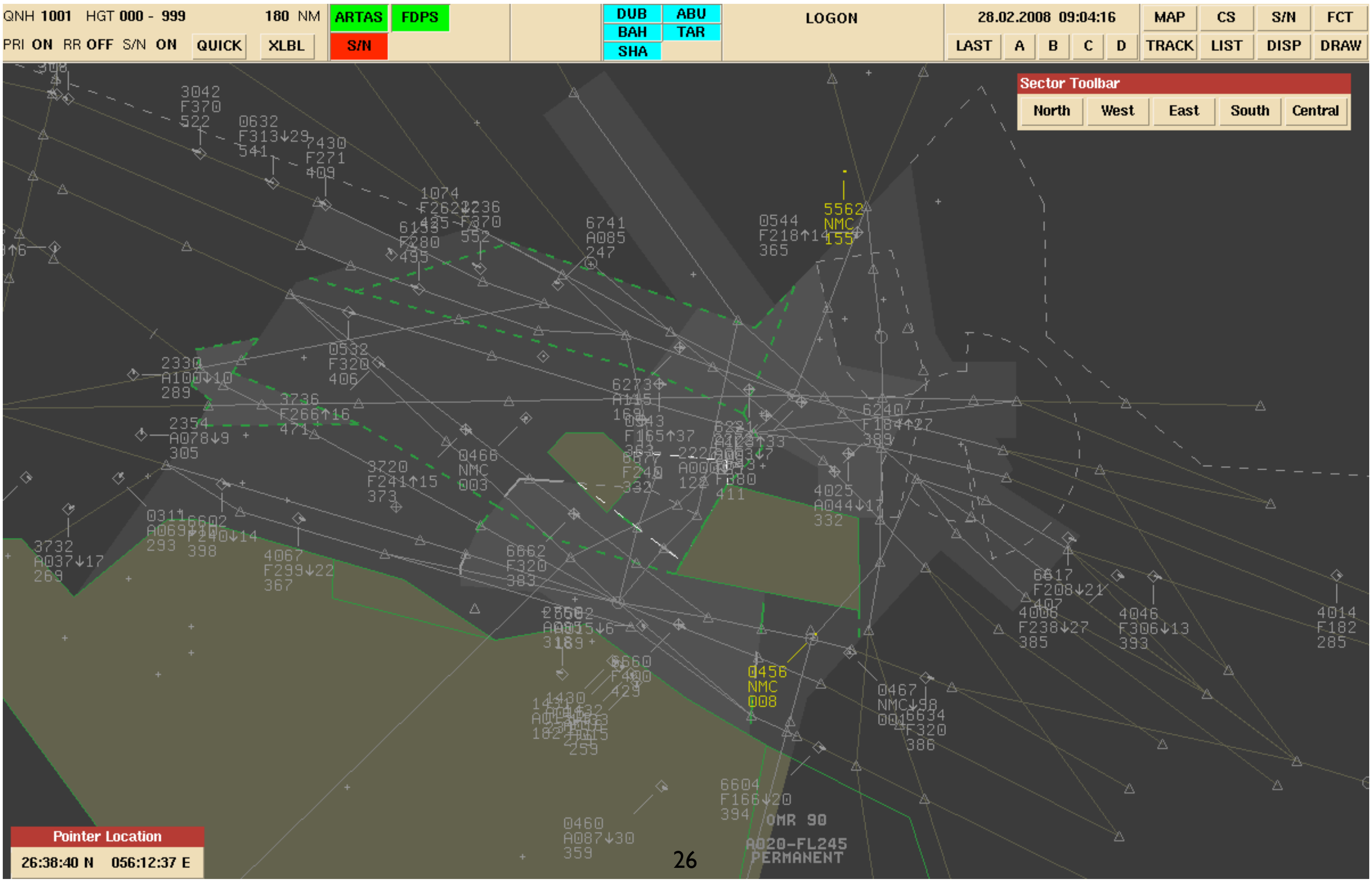
Anweisungen

Anfragen

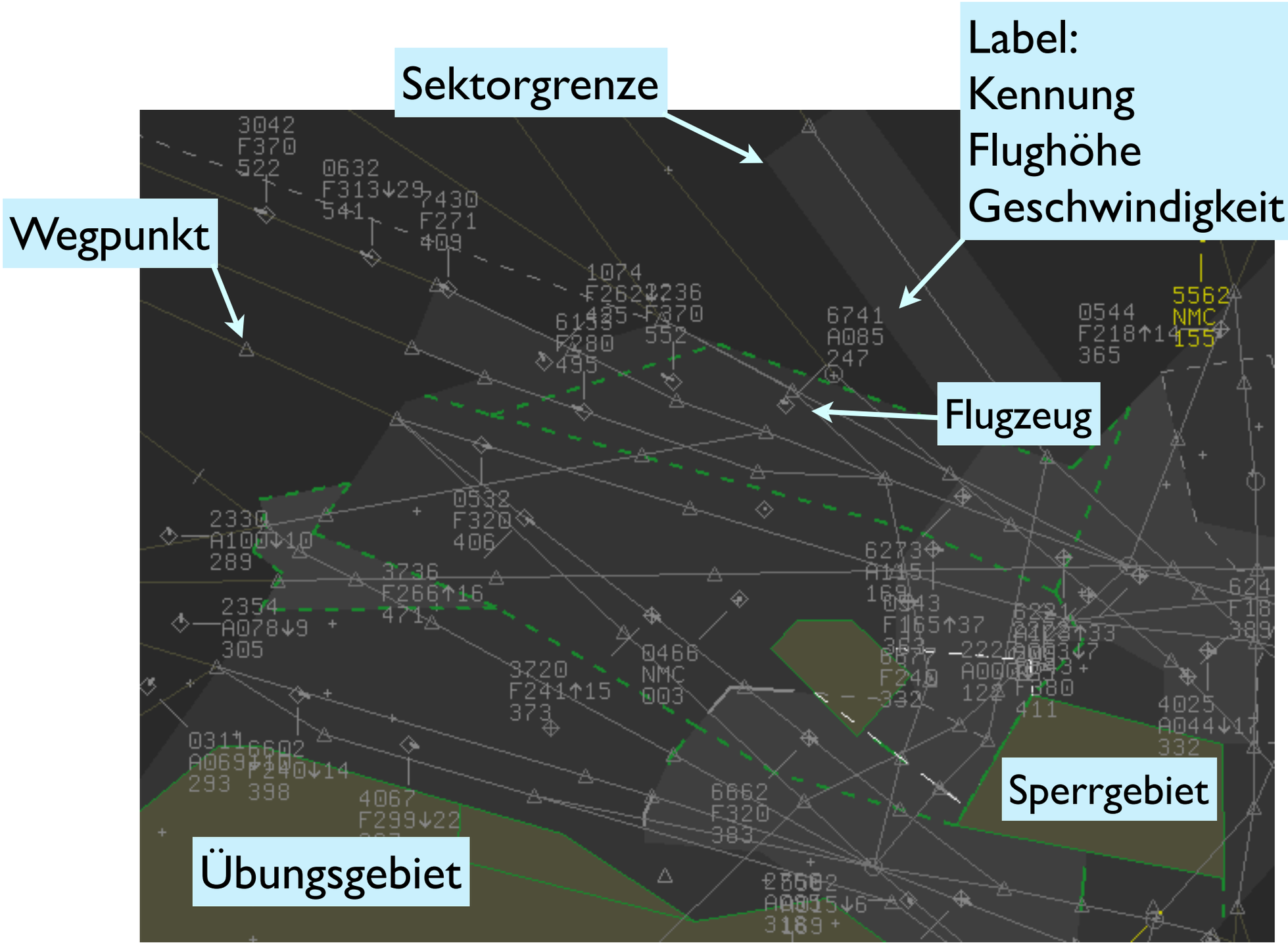


Flughafen

Luftlagebild



Luftlagebild



Filterung vermeidet Überlastung

Ziel: Fluglotse sieht nur relevanten Verkehr

Beispiel:

- Alle Flugzeuge im kontrollierten Luftraum
  - Alle Flugzeuge in der Nähe eines Flughafens
  - Ansonsten: Flugzeuge ab Flughöhe FL 100
- Ausnahme: Militärflugzeuge im Übungsgebiet

Sehr variable Forderungen der Lotsen

$\Rightarrow$

Hart-kodierte Lösungen ungeeignet

Lösung: Symbolische Logik

Filter wird durch logischen Ausdruck beschrieben

Elementarfilter (Auswertung nach Luftlage):

- Höhenband
- Id-Code (Mode-3/A) in Liste
- Geographische Filter (Polygon)

Kombinationen durch logische Operatoren

Flugzeug wird genau dann angezeigt, wenn der Filterausdruck zu “wahr” ausgewertet wird

Luftraum UAE (Beispiel)

Spezialisierte Einzelfilter:

- ∀X:a-d-app(X) ↔ inregion(X, [abu-dhabi-koord])
- ∀X: dub-app(X) ↔ inregion(X, [dubai-koord])
- ∀X: milregion(X) ↔ inregion(X, [training-koord])
- ∀X: lowairspace(X) ↔ altitude(X,0,100)
- ∀X: uppairspace(X) ↔ altitude(X,100,900)
- ∀X: military(X) ↔ modeA(X, [mil-code-list])

Luftraum UAE (II)

Filterlösung: Stelle Flugzeug X genau dann da, wenn

$$
\begin{aligned}
& ((\text{a-d-app}(X) \wedge \text{lowairspace}(X)) \\
\vee \quad & (\text{dub-app}(X) \wedge \text{lowairspace}(X)) \\
\vee \quad & \text{uppairspace}(X)) \\
\wedge \quad & (\neg\text{milregion}(X) \vee \neg\text{military}(X))
\end{aligned}
$$

mit den gegebenen Definitionen und der durch die aktuelle Luftlage definierte Interpretation zu “wahr” evaluiert wird.

Optimierung

Implementierungsdetails:

- Höhenfilter sind billig (2 Vergleiche)
- ID-Filter: Zugriff auf große Tabelle
- Geographische Filter: Teuer, sphärische Geometrie
- Auswertung erfolgt nur, wenn notwendig

Optimierte Version:

$$
\begin{array}{ll}
 & ((\text{uppairspace}(X) \\
\vee & \text{dub-app}(X) \\
\vee & \text{a-d-app}(X)) \\
\wedge & (\neg\text{military}(X) \vee \neg\text{milregion}(X))
\end{array}
$$

Gleichwertig?

$$
\begin{array}{ll}
 & ((\text{a-d-app}(X) \wedge \text{lowairspace}(X)) \\
\vee & (\text{dub-app}(X) \wedge \text{lowairspace}(X)) \\
\vee & \text{uppairspace}(X)) \\
\wedge & (\neg\text{milregion}(X) \vee \neg\text{military}(X))
\end{array}
$$

gegen

$$
\begin{array}{ll}
 & ((\text{uppairspace}(X) \\
\vee & \text{dub-app}(X) \\
\vee & \text{a-d-app}(X)) \\
\wedge & (\neg\text{military}(X) \vee \neg\text{milregion}(X))
\end{array}
$$

Verifikation

Frage: Sind ursprüngliche und optimierte Version äquivalent?

- Automatischer Beweisversuch schlägt fehl (nach 1664 Schritten/0.04 s)
- Analyse: lowairspace(X) *oder* uppairspace(X) sind die einzigen Möglichkeiten - aber das ist nicht spezifiziert!
- Mit ergänzter Spezifikation ist automatischer Beweisversuch erfolgreich (229 Schritte/0.038 s)

Fazit

Problem: Flexible Filterspezifikation mit klarer Semantik für Darstellung von Flugzeugen

Lösung: Spezifikation mit symbolischer Logik

- Mächtig
- Dynamisch konfigurierbar
- Wohlverstandene Semantik
- Automatische Verifikation möglich

Einführung Rewriting

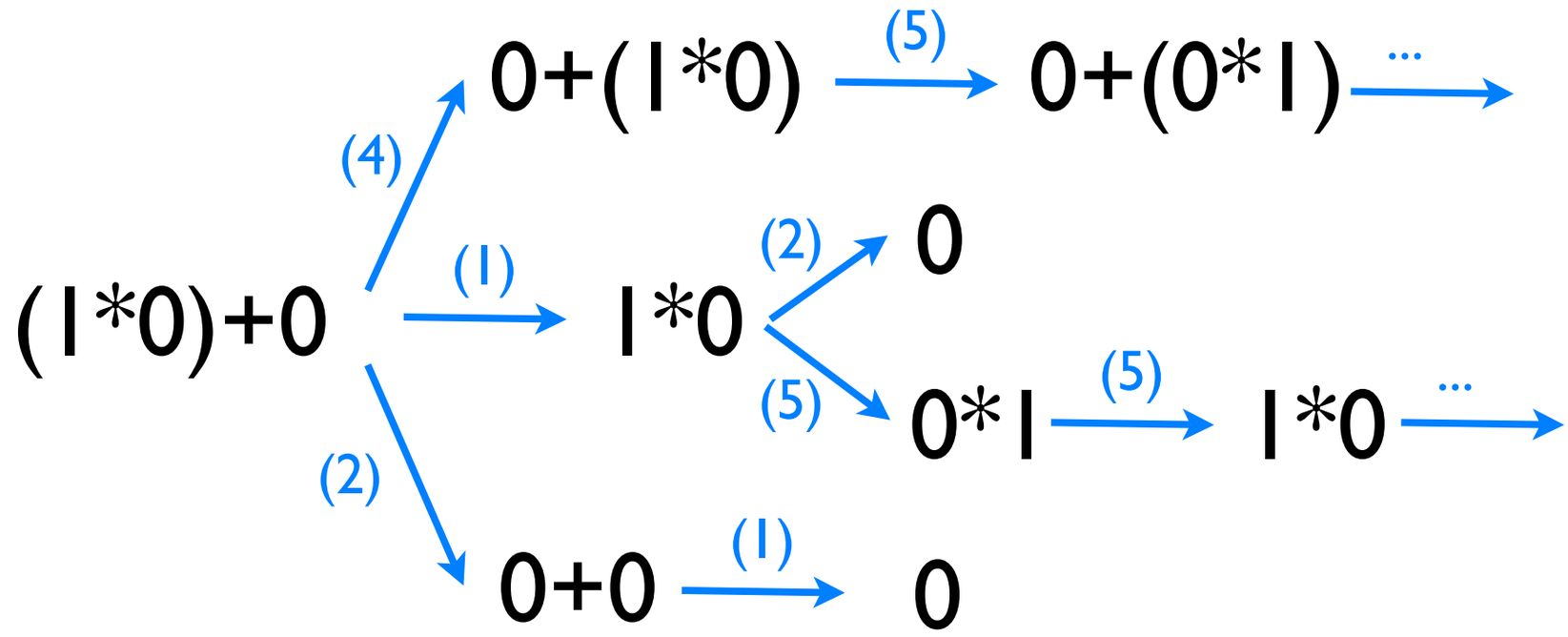
Beispiel Compiler-Optimierungen

Vereinfachung arithmetischer Ausdrücke:

Äquivalenzen:

1. x+0=0
2. x*0=0
3. x*1=x
4. x+y=y+x
5. x*y=y*x
6. x+(y+z)=(x+y)+z
7. x*(y*z)=(x*y)*z

Problem:



Beispiel fortgesetzt

Möglichkeit 1: Gleichungen dürfen beliebig eingesetzt werden

- Problem: Unendliche Ableitungen (1+2=2+1=1+2=2+1...)

Möglichkeit 2: Nur, wenn das Ergebnis "kleiner" ist

- Problem: Nicht unbedingt eindeutiges Ergebnis

Was tun?

Vervollständigung

Fakt: Wir können das Gleichungssystem vervollständigen:

```
x*0=x
x*1=x
x*0=0
0+x=x
1*x=x
0*x=0
(x+y)+z=x+(y+z)
(x*y)*z=x*(y*z)
x+y=y+x
x*y=y*x
x+(y+z)=z+(x+y)
x+(y+z)=y+(x+z)
x*(y*z)=z*(x*y)
x*(y*z)=y*(x*z)
x+(y+z)=z+(y+x)
x*(y*z)=z*(y*x)
```

Für dieses System gilt:

- Es ist äquivalent zum Ausgangssystem
- Es ist grund-konvergent
  - Also: Es gibt eindeutige Normalformen für alle Grundterme
  - Es gibt eine Ordnung, mit der das System auf Grundtermen immer terminiert

Ende der Einführung

Übungsaufgaben

Zeigen oder widerlegen Sie:

- Für alle n∈**N**$_0$ gilt: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- Die Menge aller Primzahlen ist endlich
- Es gibt keine a,b∈**N** so dass a mod b = b mod a
  - a mod b ist der ganzzahlige Divisionsrest von a/b

Hinweis: Formulieren Sie, falls nötig, geeignete Definitionen