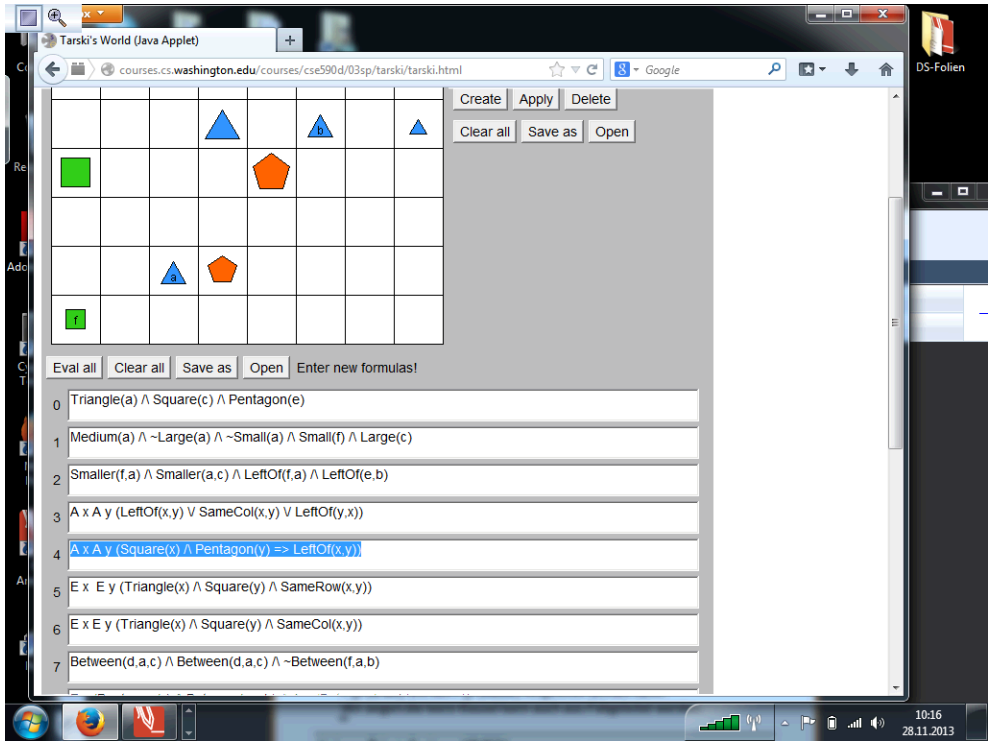
Script **generated by TTT**

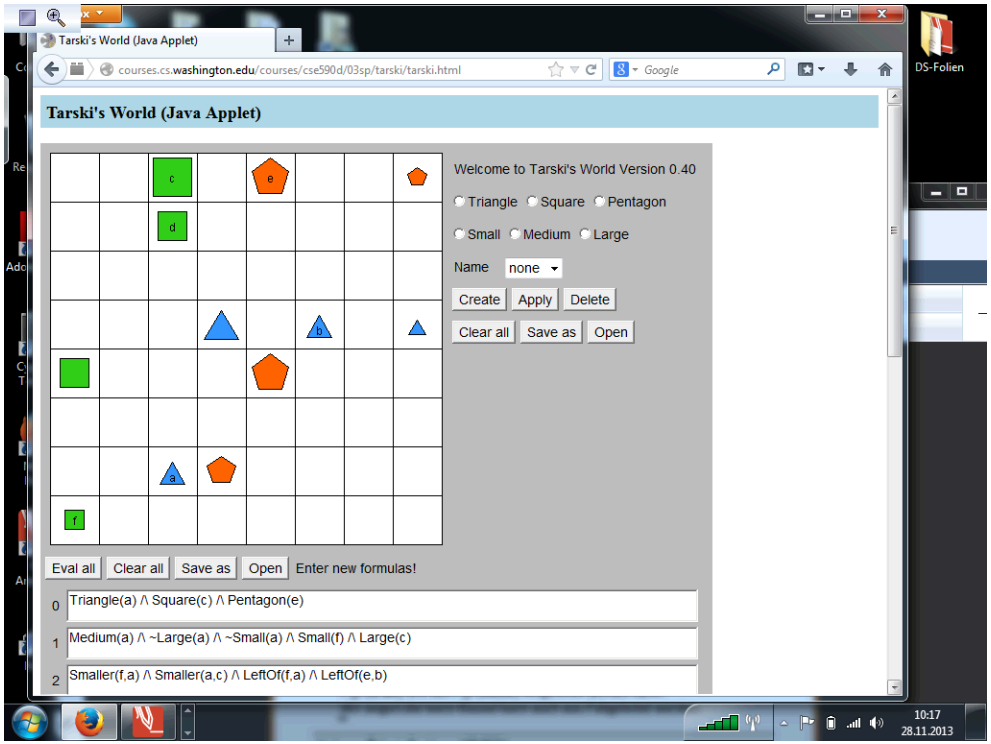
Title: Esparza: Diskrete Strukturen (28.11.2013)

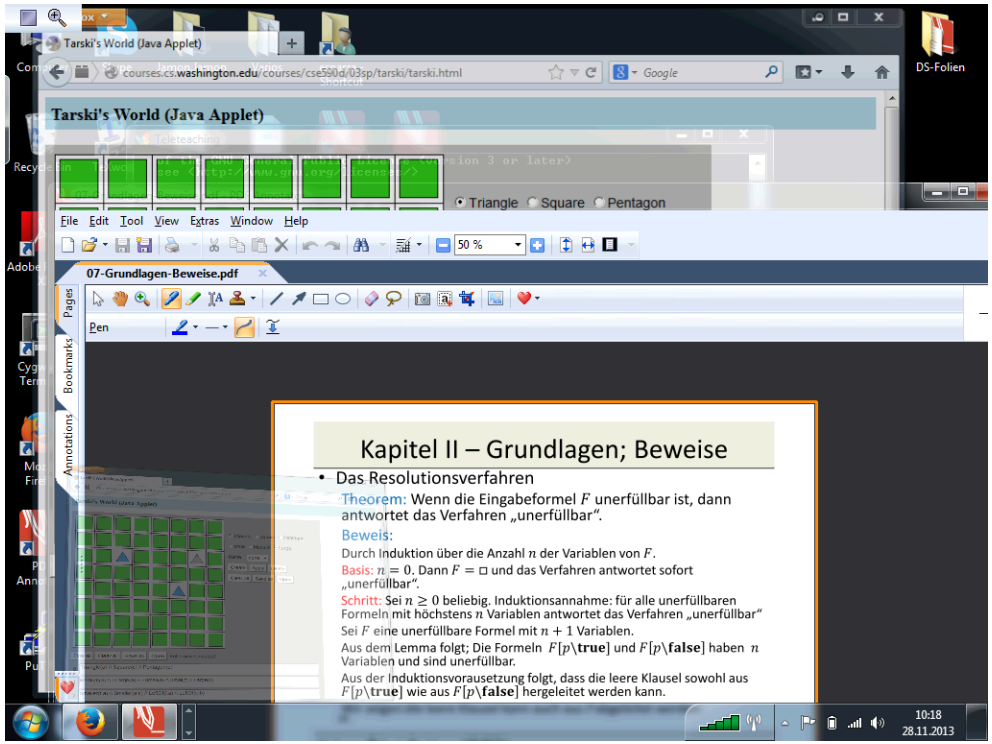
Date: Thu Nov 28 10:16:40 CET 2013

Duration: 81:08 min

Pages: 33







Kapitel II – Grundlagen; Beweise

- Transitive Hülle

Beweis (Fortsetzung):

Beweis von $\bigcup_{n\geq 1} R^n \subseteq R^+$.

Wir zeigen durch Induktion über n, dass $R^n \subseteq R^+$ für alle $n \geq 1$ gilt.

Basis. Sei $n = 1$. $R^1 \subseteq R^+$ folgt aus $R^1 = R$ und $R \subseteq R^+$.

Schritt. Sei $n \geq 1$ beliebig und nehmen wir an, dass $R^n \subseteq R^+$ gilt.

Wir zeigen $R^{n+1} \subseteq R^+$.

Sei $(x, y) \in R^{n+1}$ beliebig. Wir zeigen $(x, y) \in R^+$.

Mit $R^{n+1} = R^n \circ R$ gilt: es gibt $z \in A$ mit $(x, z) \in R^n$ und $(z, y) \in R$.

Mit $R^n \in R^+$ und $R \subseteq R^+$ gilt: es gibt $z \in A$ mit $(x, z) \in R^+$ und $(z, y) \in R^+$.

Da R^+ transitiv ist, gilt $(x, y) \in R^+$.

Kapitel II – Grundlagen; Beweise

- DPLL (Davis-Putnam-Logemann-Loveland).

Algorithmus 1:

Wenn $F = \mathbf{true}$ dann antworte „erfüllbar“

Wenn $F = \mathbf{false}$ dann antworte „unerfüllbar“

Wenn $\mathbf{true} \neq F \neq \mathbf{false}$ dann

wähle eine Variable p, die in F vorkommt;

prüfe rekursiv, ob $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ erfüllbar sind;

wenn mindestens eine von den beiden erfüllbar ist, antworte „erfüllbar“, sonst „unerfüllbar“

Kapitel II – Grundlagen; Beweise

- DPLL (Davis-Putnam-Logemann-Loveland).

Lemma. F ist erfüllbar gdw. $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ erfüllbar sind.

Beweis.

Sei β Belegung mit $[F](\beta) = 1$.

Wenn $\beta(p) = 1$ dann gilt $[F](\beta) = [F[p\backslash\mathbf{true}]](\beta) = 1$.

Wenn $\beta(p) = 0$ dann gilt $[F](\beta) = [F[p\backslash\mathbf{false}]](\beta) = 1$.

Sei β Belegung mit $[F[p\backslash\mathbf{true}]](\beta) = 1$. Sei β' die Belegung mit $\beta'(p) = 1$, und $\beta'(q) = \beta(q)$ für $q \neq p$. Dann gilt $F[\beta] = 1$.

Sei β Belegung mit $[F[p\backslash\mathbf{false}]](\beta) = 1$. Sei β' die Belegung mit $\beta'(p) = 0$, und $\beta'(q) = \beta(q)$ für $q \neq p$. Dann gilt $F[\beta] = 1$.

Kapitel II – Grundlagen; Beweise

Theorem: Das DPLL-Verfahren ist korrekt.

Beweis:

1. Das Verfahren terminiert für jede Eingabeformel F.

Durch Induktion über die Anzahl n der variablen von F.

Basis: $n = 0$. Dann $F = \mathbf{false}$ oder $F = \mathbf{true}$ und das Verfahren terminiert sofort.

Schritt: Sei $n \geq 0$ beliebig. Wir nehmen an, dass das Verfahren für alle Formeln mit n Variablen terminiert.

Sei F eine Formel mit $n + 1$ Variablen.

Die Formeln $F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ haben n Variablen.

Aus der Induktionsannahme folgt: Das Verfahren terminiert für $F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$.

So das Verfahren terminiert auch für F.

Kapitel II – Grundlagen; Beweise

Beweis (Fortsetzung):

2. Wenn F erfüllbar ist, dan antwortet das Verfahren „erfüllbar“.

Durch Induktion über die Anzahl n der variablen von F.

Basis: $n = 0$. Dann $F = \mathbf{true}$ und das Verfahren antwortet „erfüllbar“.

Schritt: Sei $n \geq 0$ beliebig. Wir nehmen an, dass für alle erfüllbaren Formeln mit höchstens n Variablen das Verfahren „erfüllbar“ antwortet.

Sei F eine erfüllbare Formel mit $n + 1$ Variablen.

Aus dem Lemma folgt: $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ sind erfüllbar.

$F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ haben höchstens n Variablen.

Aus der Induktionsannahme folgt: Das Verfahren antwortet „erfüllbar“ für $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$.

So das Verfahren anwortet „erfüllbar“.

Kapitel II – Grundlagen; Beweise

- DPLL (Davis-Putnam-Logemann-Loveland).

Lemma. F ist erfüllbar gdw. $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ erfüllbar sind.

Beweis.

Sei β Belegung mit $[F](\beta) = 1$.

Wenn $\beta(p) = 1$ dann gilt $[F](\beta) = [F[p\backslash\mathbf{true}]](\beta) = 1$.

Wenn $\beta(p) = 0$ dann gilt $[F](\beta) = [F[p\backslash\mathbf{false}]](\beta) = 1$.

Sei β Belegung mit $[F[\mathrm{p}\backslash\mathbf{true}]](\beta) = 1$. Sei β' die Belegung mit $\beta'(p) = 1$, und $\beta'(q) = \beta(q)$ für $q \neq p$. Dann gilt $F[\beta] = 1$.

Sei β Belegung mit $[F[\mathrm{p}\backslash\mathbf{false}]](\beta) = 1$. Sei β' die Belegung mit $\beta'(p) = 0$, und $\beta'(q) = \beta(q)$ für $q \neq p$. Dann gilt $F[\beta] = 1$.

Kapitel II – Grundlagen; Beweise

Beweis (Fortsetzung):

2. Wenn F erfüllbar ist, dan antwortet das Verfahren „erfüllbar“.

Durch Induktion über die Anzahl n der variablen von F.

Basis: $n = 0$. Dann $F = \mathbf{true}$ und das Verfahren antwortet „erfüllbar“.

Schritt: Sei $n \geq 0$ beliebig. Wir nehmen an, dass für alle erfüllbaren Formeln mit höchstens n Variablen das Verfahren „erfüllbar“ antwortet.

Sei F eine erfüllbare Formel mit $n + 1$ Variablen.

Aus dem Lemma folgt: $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ sind erfüllbar.

$F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ haben höchstens n Variablen.

Aus der Induktionsannahme folgt: Das Verfahren antwortet „erfüllbar“ für $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$.

So das Verfahren anwortet „erfüllbar“.

34

Vorlesung Diskrete Strukturen WS 13/14
Prof. Dr. J. Esparza – Institut für Informatik, TU München

Kapitel II – Grundlagen; Beweise

- DPLL (Davis-Putnam-Logemann-Loveland).

Algorithmus 1:

Wenn $F = \mathbf{true}$ dann antworte „erfüllbar“

Wenn $F = \mathbf{false}$ dann antworte „unerfüllbar“

Wenn $\mathbf{true} \neq F \neq \mathbf{false}$ dann

wähle eine Variable p, die in F vorkommt;

prüfe rekursiv, ob $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ erfüllbar sind;

wenn mindestens eine von den beiden erfüllbar ist, antworte „erfüllbar“, sonst „unerfüllbar“

Vorlesung Diskrete Strukturen WS 13/14
Prof. Dr. J. Esparza – Institut für Informatik, TU München

Kapitel II – Grundlagen; Beweise

Beweis (Fortsetzung):

2. Wenn F erfüllbar ist, dan antwortet das Verfahren „erfüllbar“.

Durch Induktion über die Anzahl n der variablen von F.

Basis: $n = 0$. Dann $F = \mathbf{true}$ und das Verfahren antwortet „erfüllbar“.

Schritt: Sei $n \geq 0$ beliebig. Wir nehmen an, dass für alle erfüllbaren Formeln mit höchstens n Variablen das Verfahren „erfüllbar“ antwortet.

Sei F eine erfüllbare Formel mit $n + 1$ Variablen.

Aus dem Lemma folgt: $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$ sind erfüllbar.

$F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ haben höchstens n Variablen.

Aus der Induktionsannahme folgt: Das Verfahren antwortet „erfüllbar“ für $F[p\backslash\mathbf{true}]$ oder $F[p\backslash\mathbf{false}]$.

So das Verfahren anwortet „erfüllbar“.

Kapitel II – Grundlagen; Beweise

- Das Resolutionsverfahren

while F die leere Klausel nicht enthält

{ if F zwei Klauseln K_1, K_2 enthält
mit einem Resolventen R, der $R \notin F$
erfüllt (d.h., R ist nicht Klausel von F)
then füge R als neue Klausel zu F hinzu
else antworte „erfüllbar“ und halte
}

antworte „unerfüllbar“ und halte

Kapitel II – Grundlagen; Beweise

- Das Resolutionsverfahren

Theorem: Wenn die Eingabeformel F unerfüllbar ist, dann antwortet das Verfahren „unerfüllbar“.

Beweis:

Durch Induktion über die Anzahl n der Variablen von F.

Basis: $n = 0$. Dann $F = \square$ und das Verfahren antwortet sofort „unerfüllbar“.

Schritt: Sei $n \geq 0$ beliebig. Induktionsannahme: für alle unerfüllbaren Formeln mit höchstens n Variablen antwortet das Verfahren „unerfüllbar“

Sei F eine unerfüllbare Formel mit $n + 1$ Variablen.

Aus dem Lemma folgt; Die Formeln $F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ haben n Variablen und sind unerfüllbar.

Aus der Induktionsvorausetzung folgt, dass die leere Klausel sowohl aus $F[p\backslash\mathbf{true}]$ wie aus $F[p\backslash\mathbf{false}]$ hergeleitet werden kann.

Wir zeigen:die leere Klausel kann auch aus Fabgeleitet werden.

Kapitel II – Grundlagen; Beweise

- Das Resolutionsverfahren

while F die leere Klausel nicht enthält

{ if F zwei Klauseln K_1, K_2 enthält
mit einem Resolventen R, der $R \notin F$
erfüllt (d.h., R ist nicht Klausel von F)
then füge R als neue Klausel zu F hinzu
else antworte „erfüllbar“ und halte
}

antworte „unerfüllbar“ und halte

Kapitel II – Grundlagen; Beweise

- Das Resolutionsverfahren

  Theorem: Wenn die Eingabeformel F unerfüllbar ist, dann antwortet das Verfahren „unerfüllbar“.

  Beweis:

  Durch Induktion über die Anzahl n der Variablen von F.

  Basis: $n = 0$. Dann $F = \square$ und das Verfahren antwortet sofort „unerfüllbar“.

  Schritt: Sei $n \geq 0$ beliebig. Induktionsannahme: für alle unerfüllbaren Formeln mit höchstens n Variablen antwortet das Verfahren „unerfüllbar“

  Sei F eine unerfüllbare Formel mit $n + 1$ Variablen.

  Aus dem Lemma folgt; Die Formeln $F[p\backslash\mathbf{true}]$ und $F[p\backslash\mathbf{false}]$ haben n Variablen und sind unerfüllbar.

  Aus der Induktionsvorausetzung folgt, dass die leere Klausel sowohl aus $F[p\backslash\mathbf{true}]$ wie aus $F[p\backslash\mathbf{false}]$ hergeleitet werden kann.

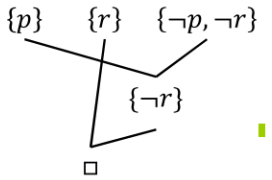
  Wir zeigen:die leere Klausel kann auch aus F abgeleitet werden.

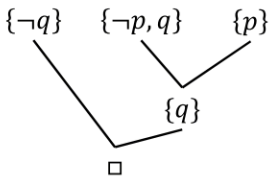
Kapitel II – Grundlagen; Beweise

$$F = \{ \{\neg q, s\}, \{\neg p, q, s\}, \{p\}, \{r, \neg s\}, \{\neg p, \neg r, \neg s\} \}$$

$$F[s\backslash\mathbf{true}] = \{ \{p\}, \{r\}, \{\neg p, \neg r\} \}$$

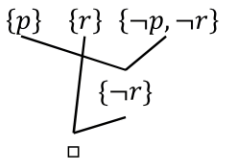
$$F[s\backslash\mathbf{false}] = \{ \{\neg q\}, \{\neg p, q\}, \{p\} \}$$

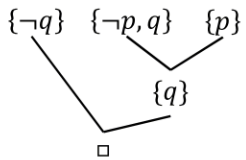


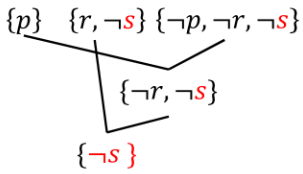


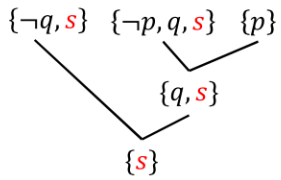
Kapitel II – Grundlagen; Beweise

$$F = \{ \{\neg q, s\}, \{\neg p, q, s\}, \{p\}, \{r, \neg s\}, \{\neg p, \neg r, \neg s\} \}$$



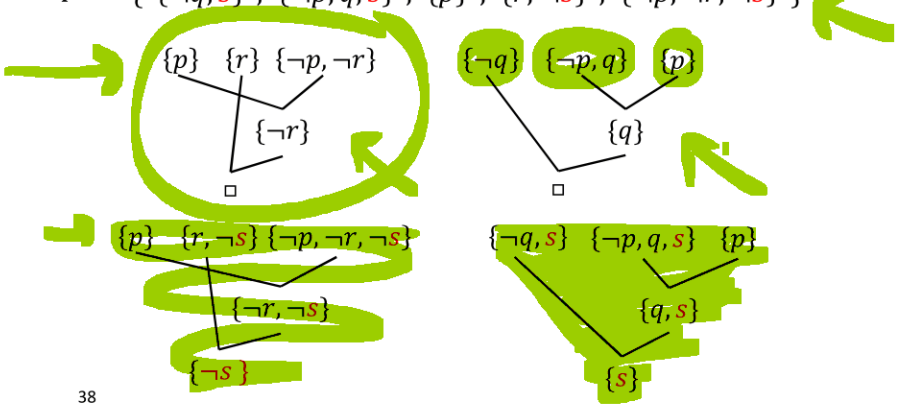




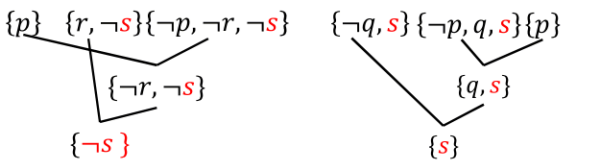


Kapitel II – Grundlagen; Beweise

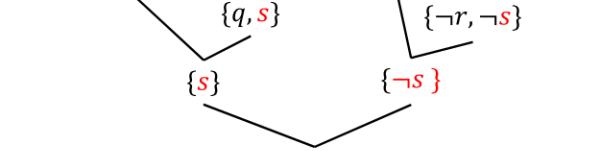
$$F = \{ \{\neg q, s\}, \{\neg p, q, s\}, \{p\}, \{r, \neg s\}, \{\neg p, \neg r, \neg s\} \}$$



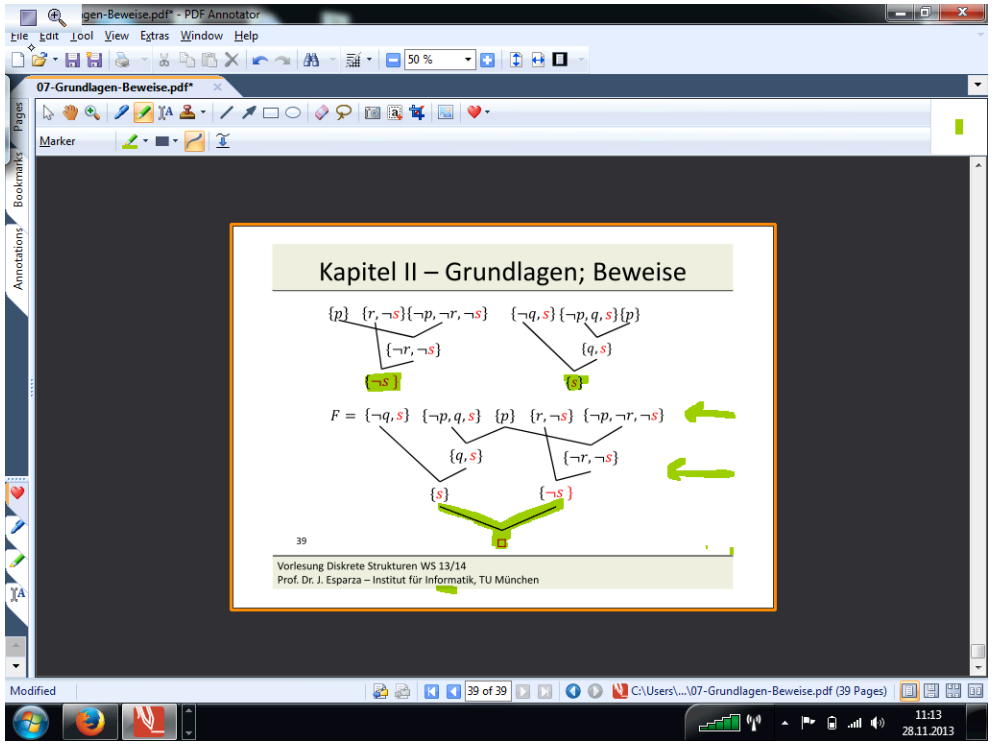
Kapitel II – Grundlagen; Beweise

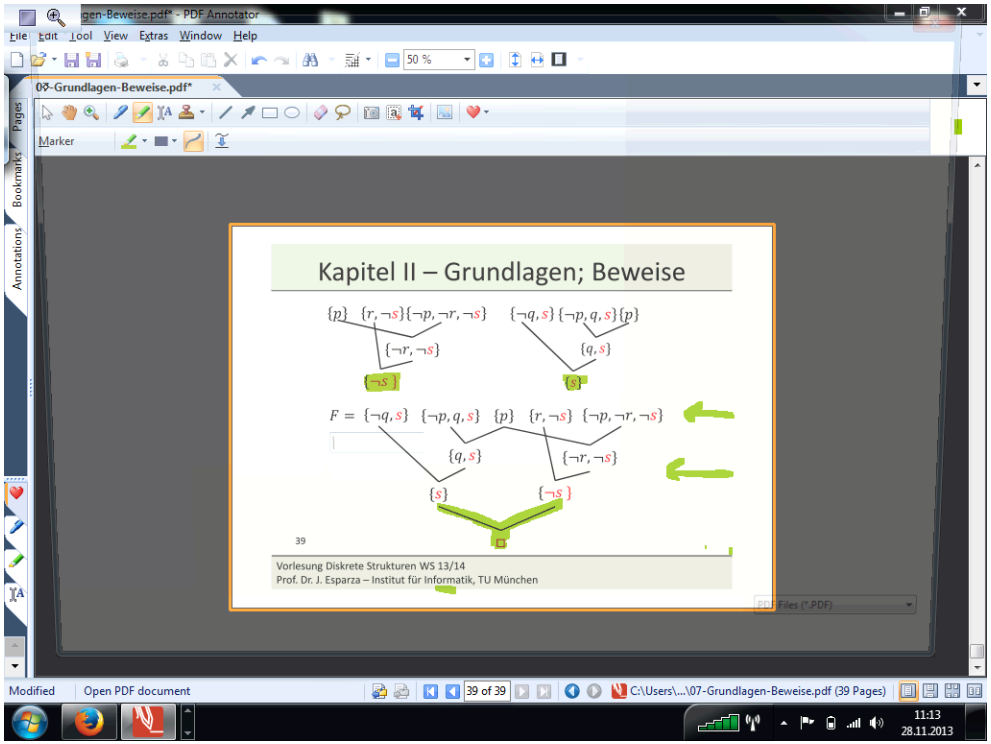


$F = \{\neg q, s\}\ \{\neg p, q, s\}\ \{p\}\ \{r, \neg s\}\ \{\neg p, \neg r, \neg s\}$



□





Kapitel II - Grundlagen

- Mathematische und notationelle Grundlagen
  - Mengen
  - Relationen und Abbildungen
  - Aussagen- und Prädikatenlogik
  - Beweismethoden
  - Wachstum von Funktionen

Kapitel II – Grundlagen; Wachstum

- Programme brauchen mehr Ressourcen (Zeit, Speicher, …) je größer ihre Eingabe
- Die Laufzeit (Speicherverbrauch) wird durch eine Funktion $f(x)$ dargestellt, mit

  $f(x)$ = maximale Laufzeit über alle Eingaben der Größe x.
- Wir sind an einer Abschätzung des Wachstums von $f(x)$ interessiert.
  - Wenn $f(x)$ schneller wächst als $g(x)$, dann wird $f(x)$ immer irgendwann größer als $g(x)$ werden (für ausreichend große Werte von x).

Kapitel II – Grundlagen; Wachstum

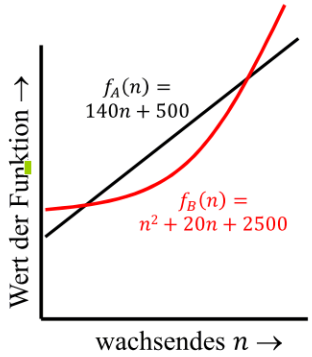
- Beispiel:
  - Eine Web-Seite soll entwickelt werden, die Benutzerdaten bearbeitet.
  - Programm A benötigt $f_A(n) = 140n + 500$ Mikrosekunden um n Einträge zu bearbeiten.
  - Programm B benötigt $f_B(n) = n^2 + 20n + 2500$ Mikrosekunden für n Einträge.

  Welches Programm soll verwendet werden?

Kapitel II – Grundlagen; Wachstum

- Eine kleine Berechung zeigt:

Für $20 \leq n \leq 100$ ist
$n^2 + 20n + 2500 \leq$
$140n + 500$



Kapitel II – Grundlagen; Wachstum

- Die multiplikativen Konstanten $140, 1, 20$, und die additiven Konstanten $500, 2500$ sind meistens schwer zu bestimmen (und abhängig von Einzelheiten der Implementierung).
- Oft ist nur bekannt:
  - Programm A braucht $an + b$ Mikrosekunden
  - Programm B braucht $cn^2 + dn + e$ Mikrosekunden

  für nicht allzu große Konstanten a,..., e.
- Wichtig ist nun: unabhängig von den Werten von $a, \ldots, e$ gibt es immer eine Zahl n_0 so dass ab n_0 Daten das Programm A schneller ist als B.

Kapitel II – Grundlagen; Wachstum

- Wir sagen, dass $f_A(n)$ langsamer als $f_B(n)$ wächst: ab einem gewissen Punkt liegt $f_A(n)$ stets unter $f_B(n)$
- Wir führen die Groß-O-Notation ein, um diesen Sachverhalt zu formalisieren. Diese wurde von Paul Bachmann (1837–1920) entwickelt, von Edmund Landau (1877–1938) verbreitet, und von D. E. Knuth in der Algorithmenanalyse eingeführt
- Das „O“ wird auch Landau Symbol genannt.
- Informell ist $O(f)$ die Menge der Funktionen, die langsamer oder so schnell wie f wachsen.

Kapitel II – Grundlagen; Wachstum

- Def. (Groß-O-Notation): Let $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x > 0\}$

$f(n) \in O(g(n))$ genau dann, wenn
$\exists c \in \mathbb{R}^+ : \exists\, n_0 \in \mathbb{N} : \forall n \geq n_0 : |f(n)| \leq c \cdot g(n)$

„f wächst (bis auf einen konstanten Faktor) höchstens so schnell wie g“

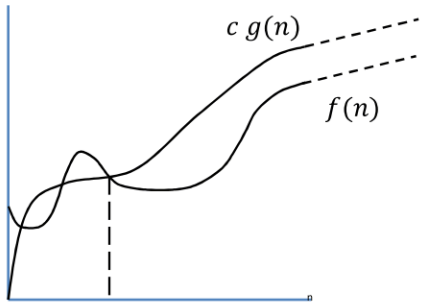
Kapitel II – Grundlagen; Wachstum

- Warum $|f(n)| \leq c \cdot g(n)$ statt $|f(n)| \leq g(n)$?
- Nehmen wir an,
  - $f(n) = an + b$
  - $g(n) = cn + d$

  für Konstanten $a, \ldots, d$.
- Wir wollen f und g als Funktionen betrachten, die genau so schnell wachsen.
- Mit der Definition auf der vorigen Folie gilt $f \in O(g)$ und $g \in O(f)$.

Kapitel II – Grundlagen; Wachstum

- Veranschaulichung der Groß-O-Notation:



Kapitel II – Grundlagen; Wachstum

- Warum $|f(n)| \leq c \cdot g(n)$ statt $|f(n)| \leq g(n)$?
- Nehmen wir an,
  - $f(n) = an + b$
  - $g(n) = cn + d$

  für Konstanten $a, \ldots, d$.
- Wir wollen f und g als Funktionen betrachten, die genau so schnell wachsen.
- Mit der Definition auf der vorigen Folie gilt $f \in O(g)$ und $g \in O(f)$.

Kapitel II – Grundlagen; Wachstum

- Beachte
  - $O(g)$ ist eine Menge von Funktionen.
  - Statt $f \in O(g)$ sagt man auch : "f ist höchstens von der Ordnung g", oder "f ist $O(g)$", oder (unsauber) "$f = O(g)$"
  - Meistens werden nur Funktionen $\mathbb{N}_0 \to \mathbb{R}^+$ betrachtet (oder $\mathbb{N} \to \mathbb{N}_0$), dann ist der Absolutbetrag überflüssig.
  - Manchmal werden auch Funktionen von $\mathbb{R} \to \mathbb{R}$ betrachtet.