Kapitel L: V

V. Produktionsregelsysteme

- Produktionsregelsysteme
- Forward-Chaining
- Backward-Chaining
- Verkettungsstrategien
- Produktionsregelsysteme mit Negation

Produktionsregelsysteme

Vergleich von Deduktions- und Produktionsregelsystem

Deduktionssysteme:

- Verwendung von Resolution bzw. einem anderen vollständigen Inferenzverfahren, um Sätze in Prädikatenlogik erster Stufe (PL1) zu beweisen.
- Beantwortung einer Anfrage geschieht durch Variableninstantiierung beim Beweis eines Satzes.
- Formeln müssen keine spezielle Form haben.

Produktionsregelsysteme

Vergleich von Deduktions- und Produktionsregelsystem

Produktionsregelsysteme:

- Zentrale Repräsentationsform ist die Implikation (Regelform) – d. h. die Formeln in der Datenbasis haben eine spezielle Form.
- Die rechte Seite einer Regel wird als Aktion interpretiert. Typische Aktionen sind die Hinzunahme und das Löschen von Fakten in einer Datenbasis sowie Ein- und Ausgabeoperationen.
- Wichtigster Schlussfolgerungsmechanismus ist die Vorwärtsverkettung. Stichwort: *Produktion*.
- Die Semantik der Regeln ist am Anwendungsbereich (Domäne) orientiert.
- Es gibt einen Konfliktauflösungsmechanismus, falls mehrere Aktionen zur Auswahl stehen.

Produktionsregelsysteme

Definition 1 (Produktionsregelsystem)

Sei Σ_P eine endliche Menge von Atomen, gebildet aus einer endlichen Menge von Objekten O_P, den Vergleichsoperatoren $\{=, \neq\}$ und einer endlichen Menge von Werten V_P.

1. Ein Atom in Σ_P hat die Form $o = v$ bzw. $o \neq v$ und wird interpretiert als „o ist gleich v“ bzw. „o ist ungleich v“.

2. $P = (D, R)$ ist ein Produktionsregelsystem; $D \subseteq \Sigma_P$ definiert die Datenbasis, R definiert eine endliche Menge von Regeln.

   Die Atome in D werden als Fakten bezeichnet.

3. Eine Regel $r \in R$ hat die Form „$\texttt{IF}\ \alpha\ \texttt{THEN}\ \tau$“. α ist eine Formel, zusammengesetzt aus Atomen aus Σ_P und den Junktoren $\wedge$ und $\vee$. τ ist ein Atom aus Σ_P.

   α wird als Bedingung oder Prämisse und τ als Konklusion der Regel bezeichnet.

Bemerkungen:

- Eine Konklusion als Konjunktion mehrerer Atome ist nicht zugelassen – jedoch

$$\texttt{IF}\ \alpha\ \texttt{THEN}\ \tau_1 \wedge \tau_2 \qquad\qquad \begin{array}{l}\texttt{IF}\ \alpha\ \texttt{THEN}\ \tau_1\\ \texttt{IF}\ \alpha\ \texttt{THEN}\ \tau_2\end{array}$$

- Die Negation in der Regel ist nicht zugelassen.

- Anstatt Objekt-Wert-Tupeln als Atome sind auch Objekt-Attribut-Wert-Tripel (OAW-Tripel) denkbar und üblich. Beispiel: EMYCIN.

- Produktionregeln können einfach um ein Konfidenzfaktorkonzept erweitert werden, das die Sicherheit von Konklusionen bewertet oder miteinander verrechnet.

Produktionsregelsysteme

Definition 2 (Semantik Produktionsregelsystem)

Eine Bedingung α ist genau dann erfüllt (wahr) bzgl. einer Datenbasis D, wenn gilt:

1. α ist ein Atom und es gilt $\alpha \in D$.
2. α hat die Form $\alpha_1 \wedge \alpha_2$ und es gilt α_1 ist wahr und α_2 ist wahr bzgl. einer Datenbasis D.
3. α hat die Form $\alpha_1 \vee \alpha_2$ und es gilt α_1 ist wahr oder α_2 ist wahr bzgl. einer Datenbasis D.
4. Nur Bedingungen, die gemäß 1 bis 3 wahr sind, sind wahr.

Eine Regel `IF` α `THEN` τ, deren Bedingung α wahr ist bzgl. einer Datenbasis D, heißt anwendbar für D.

Produktionsregelsysteme

Definition 3 (Ableitung)

Seien $P = (D, R)$ und $P' = (D', R)$ zwei Produktionsregelsysteme.

Dann gilt: „P' ist in einem Schritt aus P herleitbar“, in Zeichen: $(D, R) \mid\!\frac{1}{PS}\, (D', R)$, genau dann, wenn eine Regel $r \in R$ existiert, $r = \texttt{IF}\ \alpha\ \texttt{THEN}\ \tau$ mit

1. α ist wahr bzgl. D, d.h. r ist anwendbar für D

2. $D' = D \cup \{\tau\}$

Abkürzend: $(D, R) \mid\!\frac{1}{PS}\, \tau$

$(D, R) \mid\!\overline{}_{PS}\, (D', R)$ bezeichnet die reflexive und transitive Hülle der Einschritt-Ableitung $(D, R) \mid\!\frac{1}{PS}\, (D', R)$.

Bemerkungen:

- Eine Regel kann genau dann angewendet (gefeuert) werden, wenn die Bedingung α bzgl. der aktuellen Datenbasis erfüllt ist.
- Offensichtlich stellt die Konklusion einer gefeuerten Regel eine Folgerung dar.
- Die Wirkung einer Regelanwendung ist, dass die Konklusion τ der Regel mit in die Datenbasis aufgenommen wird.
- Der transitive Abschluss entspricht der Verkettung im Sinne der Hintereinanderausführung von Regeln.
- Ein Regelsystem ist prozedural: Implizit enthält das Feuern einer Regel eine Aktion: *„Füge Fakt zur Datenbasis hinzu“*.
- Herleitungen in einem Produktionsregelsystem sind nicht deterministisch.

Produktionsregelsysteme

Die Definition der Ableitung in Produktionsregelsystemen beschreibt den Kern eines Interpreters, den sogenannten Recognize-Act-Zyklus:

1. Bestimmung der Konfliktmenge der anwendbaren Regeln.
2. Auswahl einer Regel aus der Konfliktmenge durch ein Selektionsverfahren.
3. Feuern der Regel.

Insbesondere legt der Interpreter fest, wie die Konfliktmenge gebildet werden kann und aus welchen Kriterien das Selektionsverfahren aufgebaut ist.

Produktionsregelsysteme

Die Definition der Ableitung in Produktionsregelsystemen beschreibt den Kern eines Interpreters, den sogenannten Recognize-Act-Zyklus:

1. Bestimmung der Konfliktmenge der anwendbaren Regeln.
2. Auswahl einer Regel aus der Konfliktmenge durch ein Selektionsverfahren.
3. Feuern der Regel.

Insbesondere legt der Interpreter fest, wie die Konfliktmenge gebildet werden kann und aus welchen Kriterien das Selektionsverfahren aufgebaut ist.

Zwei Möglichkeiten, um einen Finalzustand zu ereichen:

1. Alle herleitbaren Fakten sind abgeleitet; keine Regel mehr anwendbar.
   Paradigma: „Finde soviel wie möglich heraus."
2. Ein gesuchter Fakt ist zu D hinzugefügt worden.
   Paradigma: „Ist ein bestimmtes Ziel folgerbar?"

Produktionsregelsysteme

Definition 4 (kommutativ)

Ein Produktionsregelsystem $P = (D, R)$ heißt kommutativ, falls für jede Datenbasis D_i, die aus P ableitbar ist, gilt:

Eine für D_i anwendbare Regel ist auch für jede Datenbasis D_i' anwendbar, für die $(D_i, R) \vdash_{PS} (D_i', R)$ gilt.

Produktionsregelsysteme

Definition 4 (kommutativ)

Ein Produktionsregelsystem $P = (D, R)$ heißt kommutativ, falls für jede Datenbasis D_i, die aus P ableitbar ist, gilt:

Eine für D_i anwendbare Regel ist auch für jede Datenbasis D_i' anwendbar, für die $(D_i, R) \mid\!\underline{}_{PS}\, (D_i', R)$ gilt.

Lemma 1

Für ein kommutatives Produktionsregelsystem $P = (D, R)$ und zwei Datenbasen D_1, D_2, die aus P ableitbar sind, gilt die folgende Eigenschaft:

Sei D_1' eine Datenbasis, die aus D_1 ableitbar ist, so existiert eine Folge von Regelanwendungen, um $D_1' \cup D_2$ aus D_2 abzuleiten. Die Generierung der Fakten in D_1' ist also unabhängig von der Anwendungsreihenfolge der anwendbaren Regeln.

Satz 1

Produktionsregelsysteme ohne Negation sind kommutativ.

Produktionsregelsysteme

Realisierung des Interpreters durch Regelverkettung:

$$\cfrac{D_0,\ \texttt{IF}\ \alpha_1\ \texttt{THEN}\ \tau_1 \quad \text{(und } \alpha_1 \text{ wahr bzgl. } D_0\text{)}}{\cfrac{D_1 = D_0 \cup \{\tau_1\},\ \texttt{IF}\ \alpha_2\ \texttt{THEN}\ \tau_2 \quad \text{(und } \alpha_2 \text{ wahr bzgl. } D_1\text{)}}{\cfrac{D_2 = D_1 \cup \{\tau_2\},\ \texttt{IF}\ \alpha_3\ \texttt{THEN}\ \tau_3 \quad \text{(und } \alpha_3 \text{ wahr bzgl. } D_2\text{)}}{\cfrac{D_3 = D_2 \cup \{\tau_3\},\ \ldots}{\ddots}}}}$$

Die Kommutativität wird hier insofern ausgenutzt, als dass die Reihenfolge der Regelanwendungen keinen Einfluss auf die Menge der abgeleiteten Fakten hat.

Produktionsregelsysteme

Realisierung des Interpreters durch Regelverkettung:

$$\cfrac{D_0,\ \texttt{IF}\ \alpha_1\ \texttt{THEN}\ \tau_1}{\cfrac{D_1 = D_0 \cup \{\tau_1\},\ \texttt{IF}\ \alpha_2\ \texttt{THEN}\ \tau_2}{\cfrac{D_2 = D_1 \cup \{\tau_2\},\ \texttt{IF}\ \alpha_3\ \texttt{THEN}\ \tau_3}{\cfrac{D_3 = D_2 \cup \{\tau_3\},\ \ldots}{\ddots}}}}$$

(und α_1 wahr bzgl. D_0)

(und α_2 wahr bzgl. D_1)

(und α_3 wahr bzgl. D_2)

Die Kommutativität wird hier insofern ausgenutzt, als dass die Reihenfolge der Regelanwendungen keinen Einfluss auf die Menge der abgeleiteten Fakten hat.

Vergleiche Modus Ponens in der Logik ($\alpha, \beta, \gamma, \delta, \ldots$ beliebige logische Formeln) :

$$\cfrac{\alpha, \quad \alpha \to \beta}{\cfrac{\beta, \quad \beta \to \gamma}{\cfrac{\gamma, \quad \gamma \to \delta}{\cfrac{\delta, \quad \ldots}{\ddots}}}}$$

Produktionsregelsysteme

vorwärtsverkettende Verfahren (Forward-Chaining)

Ausgehend von D_0 wird versucht, einen gegebenen Fakt τ_i bzw. die Menge aller ableitbaren Fakten zu abzuleiten. Stichwort: *datengetriebene Suche*

$$\underline{D_0,\ \texttt{IF}\ \alpha_1\ \texttt{THEN}\ \tau_1}$$

$$\downarrow \qquad \underline{D_1 = D_0 \cup \{\tau_1\},\ \texttt{IF}\ \alpha_2\ \texttt{THEN}\ \tau_2 \ldots}$$

$$\underline{D_2 = D_1 \cup \{\tau_2\},\ \ldots}$$

$$\ddots$$

Produktionsregelsysteme

vorwärtsverkettende Verfahren (Forward-Chaining)

Ausgehend von D_0 wird versucht, einen gegebenen Fakt τ_i bzw. die Menge aller ableitbaren Fakten zu abzuleiten. Stichwort: *datengetriebene Suche*

$$\underline{D_0,\ \texttt{IF}\ \alpha_1\ \texttt{THEN}\ \tau_1}$$

$$\underline{D_1 = D_0 \cup \{\tau_1\},\ \texttt{IF}\ \alpha_2\ \texttt{THEN}\ \tau_2 \ldots}$$

$$\underline{D_2 = D_1 \cup \{\tau_2\},\ \ldots}$$

$$\ddots$$

rückwärtsverkettende Verfahren (Backward-Chaining)

Ausgehend von einem zu bestimmenden Fakt τ_i wird versucht, diesen über Regeln auf eine Teilmenge der vorhandenen Startdatenbasis D_0 zurückzuführen. Stichwort: *zielgetriebene Suche*

$$\underline{D_0,\ \texttt{IF}\ \alpha_1\ \texttt{THEN}\ \tau_1}$$

$$\underline{D_1 = D_0 \cup \{\tau_1\},\ \texttt{IF}\ \alpha_2\ \texttt{THEN}\ \tau_2 \ldots}$$

$$\underline{D_2 = D_1 \cup \{\tau_2\},\ \ldots}$$

$$\ddots$$

Forward-Chaining

Recognize-Act-Zyklus realisiert als Forward-Chaining:

1. Recognize: Konstruktion der Konfliktmenge R^*
   Bestimmung aller Regeln, deren Bedingung wahr ist.

2. Act: Feuern einer Regel
   Auswahl einer Regel aus R^* und Ausführung ihrer Konklusion.

Algorithm: FC
Input: Startdatenbasis D, Regelmenge R
Output: Menge aller aus D mit R ableitbaren Fakten D^*

```
BEGIN
  D* = D

  REPEAT
    D_tmp = D*
    R* = {(IF α THEN τ) ∈ R | α wahr bzgl. D*}
    D* = D* ∪ {τ | (IF α THEN τ) ∈ R*}
  UNTIL D* = D_tmp

  RETURN (D*)
END
```

Forward-Chaining

Eigenschaften des Algorithmus FC:

- FC terminiert bei jeder Eingabe.

  Die Größe von D^* ist beschränkt durch die endliche Menge der möglichen Atome in P.

- FC bestimmt genau die Menge aller ableitbaren Fakten.

  Beweis über die Kommutativität von P.

- FC benötigt höchstens quadratische Zeit in der Größe von $P = (D, R)$.

  Lineare Zeit für jeden Schleifendurchlauf (falls Test, ob ein Fakt für D^* wahr ist, sowie das Hinzufügen von Fakten in einem Schritt möglich sind); die Größe von D^* bestimmt die Anzahl der Schleifendurchläufe.

Forward-Chaining

Algorithm: FC-test. Überprüft, ob ein Atom τ^* ableitbar ist.

Input: Startdatenbasis D, Regelmenge R, Atom τ^*

Output: *true*, falls $(D, R) \mid_{\overline{PS}} \tau^*$, *false* sonst

```
BEGIN
  D* = D

  REPEAT
    D_tmp = D*
    R* = {(IF α THEN τ) ∈ R | α wahr bzgl. D*}
    D* = D* ∪ {τ | (IF α THEN τ) ∈ R*}
  UNTIL D* = D_tmp OR τ* ∈ D*

  IF τ* ∈ D*
  THEN RETURN (true)
  ELSE RETURN (false)
END
```

Forward-Chaining

Algorithm: FC-test. Überprüft, ob ein Atom τ^* ableitbar ist.

Input: Startdatenbasis D, Regelmenge R, Atom τ^*

Output: *true*, falls $(D, R) \mid_{\overline{PS}} \tau^*$, *false* sonst

```
BEGIN
  D* = D

  REPEAT
    D_tmp = D*
    R* = {(IF α THEN τ) ∈ R | α wahr bzgl. D*}
    D* = D* ∪ {τ | (IF α THEN τ) ∈ R*}
  UNTIL D* = D_tmp OR τ* ∈ D*

  IF τ* ∈ D*
  THEN RETURN (true)
  ELSE RETURN (false)
END
```

Bemerkung:

Eine Verbesserung der Effzienz ist dadruch möglich, dass Regeln, die gefeuert haben, aus der Konfliktmenge entfernt werden.

- Warum bleibt Korrektheit?
- Wie verhält sich die Laufzeit?

Backward-Chaining

Recognize-Act-Zyklus realisiert als Backward-Chaining:

1. Recognize: Konstruktion der Konfliktmenge R^*

   Bestimmung aller Regeln, die das zu prüfende Atom als Konklusion haben, bzw. Prüfung, ob das Atom in der Startdatenbasis enthalten ist.

2. Act: Feuern einer Regel

   Auswahl einer bestimmten Regel aus R^* und Generierung neuer Ziele aus der Bedingung α dieser Regel.

Situationen mit Nichtdeterminismen:

- Eventuell existieren mehrere Regeln mit der Konklusion τ.
- Die Bedingung kann zusammengesetzt sein – dann ist die Reihenfolge der Bearbeitung entscheidend:

  Konjunktion: Welche Teilformel ist nicht ableitbar?

  Disjunktion: Welche Teilformel ist (schnell) ableitbar?

Backward-Chaining

Konstruktion eines Und-Oder-Baums $AOT_P(G)$ (And-Or-Tree) zu einem Produktionsregelsystem $P = (D, R)$ und einem Ziel G.

Definition 5 (Und-Oder-Baum)

1. Die Wurzel von $AOT_P(G)$ erhält den Label G.

2. Ist der Label eines Knotens ein Atom, so erhält der Knoten einen Nachfolger
   - mit Label $\Box$, falls $\tau \in D$,
   - mit Label α für jede Regel `IF` α `THEN` τ in R.

   Die Kanten zu den Nachfolgern sind vom Typ ODER.

3. Hat ein Knoten einen Label mit der Struktur $\alpha_1 \wedge \ldots \wedge \alpha_n$, so erhält der Knoten n Nachfolger mit den Labeln α_1 bis α_n.
   Die Kanten zu den Nachfolgern sind vom Typ UND.

4. Hat ein Knoten einen Label mit der Struktur $\alpha_1 \vee \ldots \vee \alpha_n$, so erhält der Knoten n Nachfolger mit den Labeln α_1 bis α_n.
   Die Kanten zu den Nachfolgern sind vom Typ ODER.

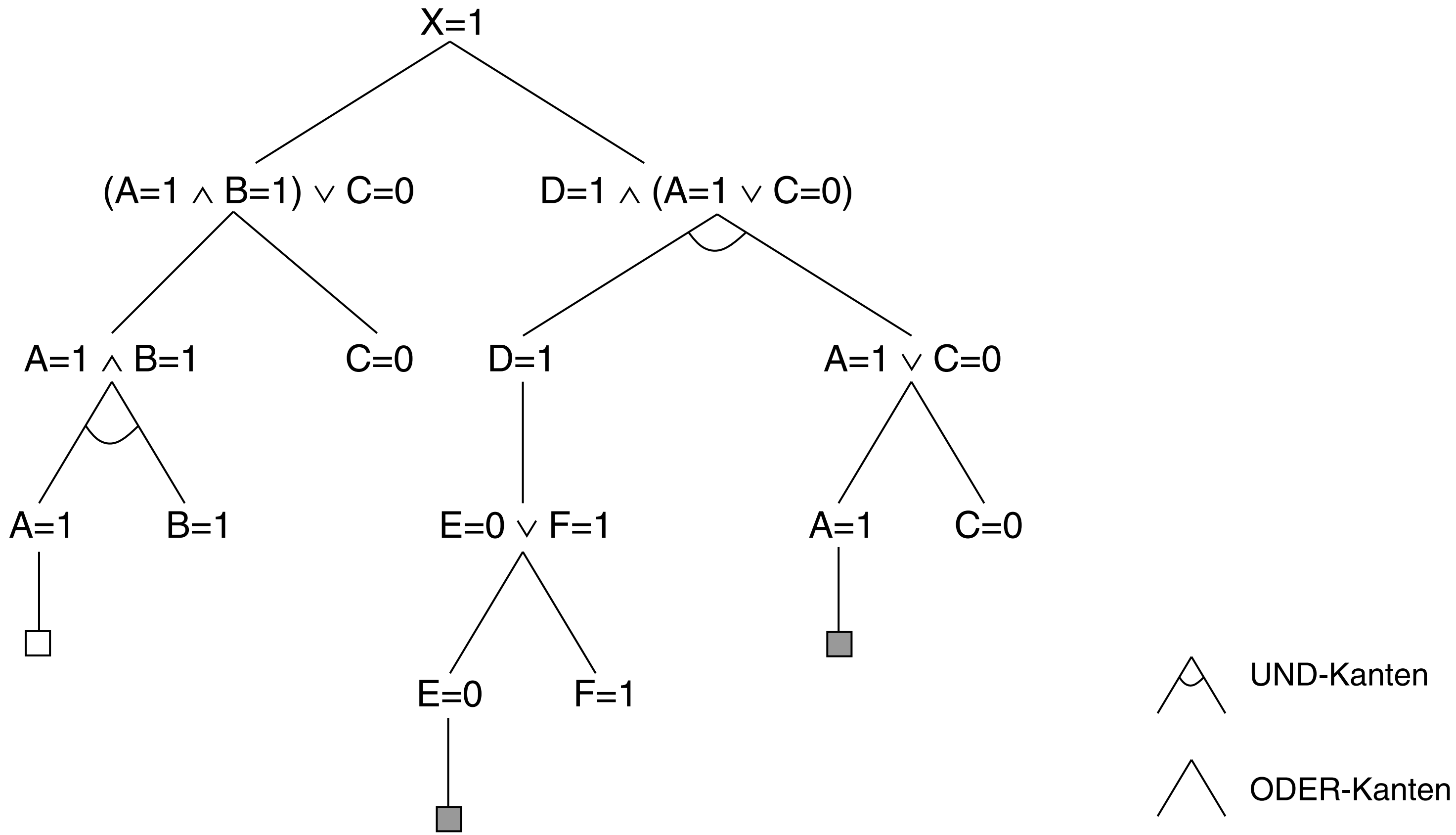
5. $AOT_P(G)$ enthält keine anderen Knoten und Kanten.

Backward-Chaining

Beispiel Und-Oder-Baum. Gegeben sei folgendes Produktionsregelsystem $P = (D, R)$:

$D = \{A = 1, E = 0\}$ $\quad R = \{r_1 : \texttt{IF } (A = 1 \wedge B = 1) \vee C = 0 \texttt{ THEN } X = 1,$
$\quad r_2 : \texttt{IF } D = 1 \wedge (A = 1 \vee C = 0) \texttt{ THEN } X = 1,$
$\quad r_3 : \texttt{IF } E = 0 \vee F = 1 \texttt{ THEN } D = 1\}$

Für Ziel $X = 1$:



Backward-Chaining

Algorithm: BC-DFS

Input: Startdatenbasis D, Regelmenge R, Formel α

Output: *true*, falls α ableitbar, *false* sonst; evtl. Endlosschleife

```
BEGIN
  IF α = α1 ∧ α2 THEN RETURN (BC-DFS(α1) AND BC-DFS(α2)) ENDIF
  IF α = α1 ∨ α2 THEN RETURN (BC-DFS(α1) OR BC-DFS(α2)) ENDIF
  IF α ∈ D THEN RETURN (true) ENDIF
```

Backward-Chaining

Algorithm: BC-DFS
Input: Startdatenbasis D, Regelmenge R, Formel α
Output: *true*, falls α ableitbar, *false* sonst; evtl. Endlosschleife

```
BEGIN
  IF α = α1 ∧ α2 THEN RETURN (BC-DFS(α1) AND BC-DFS(α2)) ENDIF
  IF α = α1 ∨ α2 THEN RETURN (BC-DFS(α1) OR BC-DFS(α2)) ENDIF
  IF α ∈ D THEN RETURN (true) ENDIF

  R* = {r | r = (IF γ THEN α) und r ∈ R}
  stop=false

  WHILE R* ≠ ∅ AND stop=false DO
      r = choose(R*)
      IF BC-DFS(premise(r))=true
      THEN stop=true
      ELSE R* = R* \ {r}
  END

  IF stop=true
  THEN RETURN (true)
  ELSE RETURN (false)

END
```

Backward-Chaining

Bedingungen ohne Disjunktion

Im UND-ODER-Baum wird nicht zwischen alternativen Regeln und Disjunktionen unterschieden.

- $\Rightarrow$ Die ausschließliche Verwendung von Konjunktionen ist ohne Einschränkung hinsichtlich der Ausdrucksstärke.
- $\Rightarrow$ Konstruktion eines Ableitungsbaums, der nur Konjunktionen enthält.
- $\Rightarrow$ Aufspaltung von Regeln mit Disjunktion (Fortsetzung Beispiel):

$$\begin{aligned}
r_1 &: \texttt{IF } (A = 1 \wedge B = 1) \vee C = 0 \texttt{ THEN } X = 1,\\
r_2 &: \texttt{IF } D = 1 \wedge (A = 1 \vee C = 0) \texttt{ THEN } X = 1,\\
r_3 &: \texttt{IF } E = 0 \vee F = 1 \texttt{ THEN } D = 1
\end{aligned}$$

Aus r_1 wird:

$$\begin{aligned}
r_{1.1} &: \texttt{IF } A = 1 \wedge B = 1 \texttt{ THEN } X = 1\\
r_{1.2} &: \texttt{IF } C = 0 \texttt{ THEN } X = 1
\end{aligned}$$

Aus r_3 wird:

$$\begin{aligned}
r_{3.1} &: \texttt{IF } E = 0 \texttt{ THEN } D = 1\\
r_{3.2} &: \texttt{IF } F = 1 \texttt{ THEN } D = 1
\end{aligned}$$

Backward-Chaining

Bedingungen ohne Disjunktion (Fortsetzung)

Zwei Möglichkeiten, um r_2 umzuformen:

1. Einführung von Atomen *aux* $= 1$, die bisher nicht in P nicht existieren.

$$\begin{aligned} r_{2.1} &: \texttt{IF } D = 1 \wedge \textit{aux} = 1 \texttt{ THEN } X = 1 \\ r_{2.2} &: \texttt{IF } A = 1 \texttt{ THEN } \textit{aux} = 1, \\ r_{2.3} &: \texttt{IF } C = 0 \texttt{ THEN } \textit{aux} = 1 \end{aligned}$$

2. Erzeugung der disjunktiven Normalform durch iterative Anwendung der Distributivgesetze:

$$\begin{aligned} t \wedge (t_1 \vee \ldots \vee t_n) &\approx (t \wedge t_1) \vee \ldots \vee (t \wedge t_n) \\ t \vee (t_1 \wedge \ldots \wedge t_n) &\approx (t \vee t_1) \wedge \ldots \wedge (t \vee t_n) \end{aligned}$$

$\leadsto$

$$\begin{aligned} r_{2.1} &: \texttt{IF } D = 1 \wedge A = 1 \texttt{ THEN } X = 1 \\ r_{2.2} &: \texttt{IF } D = 1 \wedge C = 0 \texttt{ THEN } X = 1 \end{aligned}$$

Backward-Chaining

Konstruktion eines Ableitungsbaums $T_P(G)$ zu einem Produktionsregelsystem $P = (D, R)$ und einem Ziel G.

Definition 6 (Ableitungsbaum)

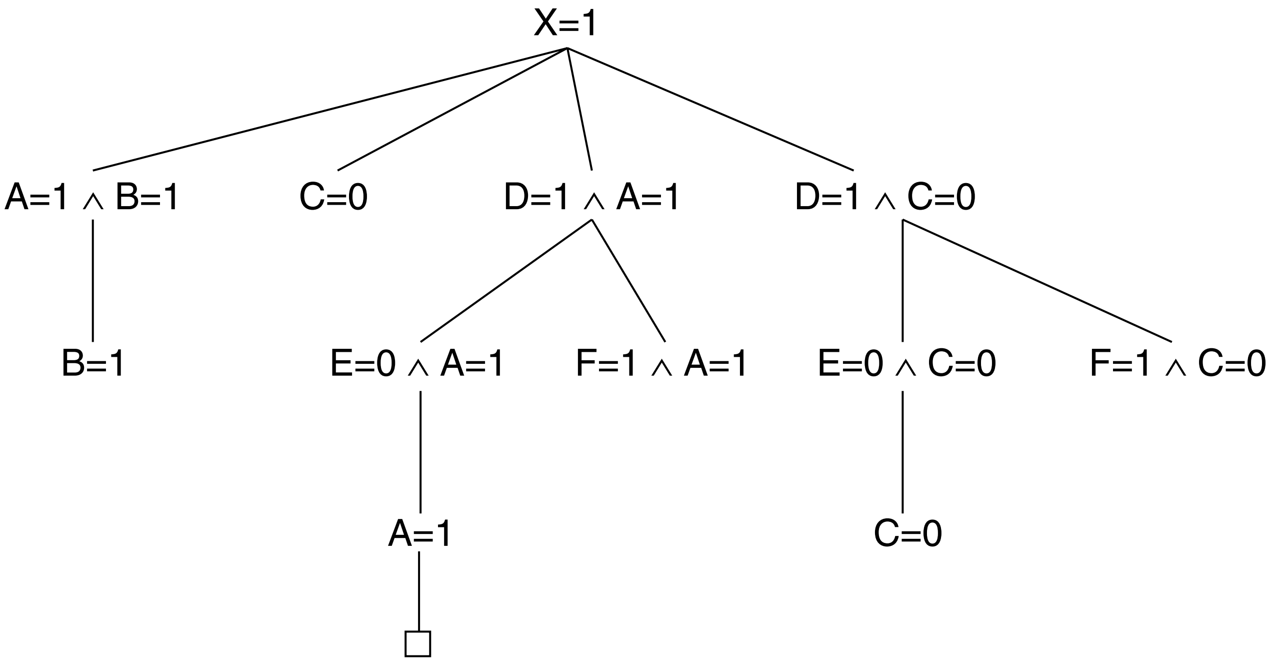
Siehe Übung.

Backward-Chaining

Beispiel Ableitungsbaum. Gegeben sei folgendes Produktionsregelsystem $P = (D, R)$:

$$D = \{A = 1, E = 0\} \qquad R = \{\begin{array}{l} r_{1.1} : \texttt{IF } A = 1 \land B = 1 \texttt{ THEN } X = 1 \\ r_{1.2} : \texttt{IF } C = 0 \texttt{ THEN } X = 1 \\ r_{2.1} : \texttt{IF } D = 1 \land A = 1 \texttt{ THEN } X = 1 \\ r_{2.2} : \texttt{IF } D = 1 \land C = 0 \texttt{ THEN } X = 1 \\ r_{3.1} : \texttt{IF } E = 0 \texttt{ THEN } D = 1 \\ r_{3.2} : \texttt{IF } F = 1 \texttt{ THEN } D = 1 \end{array}$$

Für Ziel $X = 1$:



Backward-Chaining

Die Analyse von BC-DFS erfordert die Analyse eines Regelgraphen.

Definition 7 (Regelgraph)

Sei R eine Regelmenge ohne Disjunktionen. Ein Regelgraph $G_R = \langle V, E \rangle$ ist ein gerichteter Graph, der wie folgt definiert ist:

1. Für jedes in R vorkommende Atom τ existiert ein Knoten v_τ in V.
2. Für jede Regel $r \in R$ existiert ein Knoten v_r in V.
3. Für jede Regel $r = \texttt{IF}\ \alpha_1 \wedge \ldots \wedge \alpha_n\ \texttt{THEN}\ \tau$ existieren n Kanten von v_{α_i} nach v_r $(i = 1, \ldots, n)$ und eine Kante von v_r nach v_τ in E.
4. G_R enthält keine anderen Knoten und Kanten.

Backward-Chaining

Die Analyse von BC-DFS erfordert die Analyse eines Regelgraphen.

Definition 7 (Regelgraph)

Sei R eine Regelmenge ohne Disjunktionen. Ein Regelgraph $G_R = \langle V, E \rangle$ ist ein gerichteter Graph, der wie folgt definiert ist:

1. Für jedes in R vorkommende Atom τ existiert ein Knoten v_τ in V.
2. Für jede Regel $r \in R$ existiert ein Knoten v_r in V.
3. Für jede Regel $r = \texttt{IF}\ \alpha_1 \wedge \ldots \wedge \alpha_n\ \texttt{THEN}\ \tau$ existieren n Kanten von v_{α_i} nach v_r $(i = 1, \ldots, n)$ und eine Kante von v_r nach v_τ in E.
4. G_R enthält keine anderen Knoten und Kanten.

Definition 8 (zyklenfreie Regelmenge)

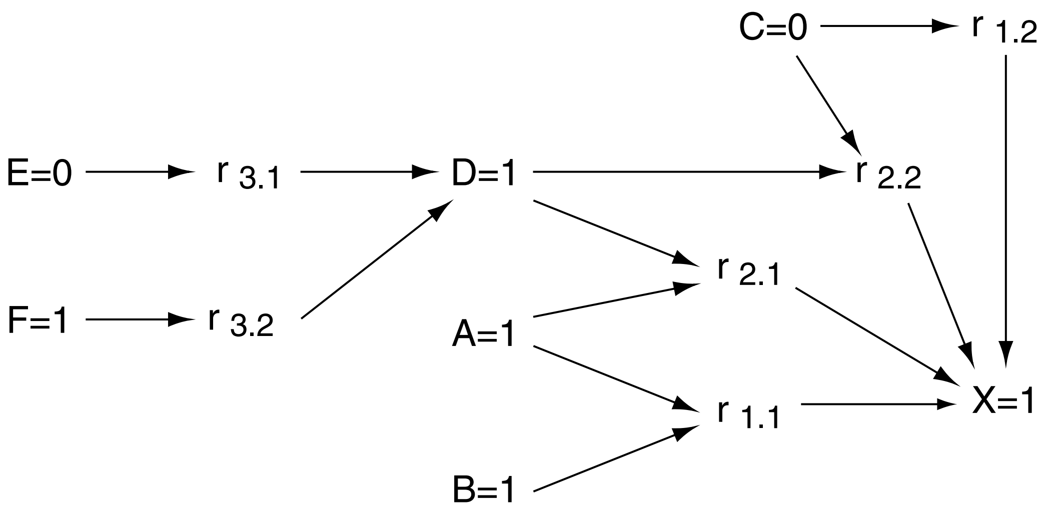
Eine Regelmenge R heißt zyklenfrei, wenn der zugehörige Regelgraph G_R keine Zyklen enthält.

Backward-Chaining

Beispiel Regelgraph. Gegeben sei folgende Regelmenge:

$$
\begin{aligned}
R = \{ r_{1.1} &: \texttt{IF } A = 1 \wedge B = 1 \texttt{ THEN } X = 1 \\
r_{1.2} &: \texttt{IF } C = 0 \texttt{ THEN } X = 1 \\
r_{2.1} &: \texttt{IF } D = 1 \wedge A = 1 \texttt{ THEN } X = 1 \\
r_{2.2} &: \texttt{IF } D = 1 \wedge C = 0 \texttt{ THEN } X = 1 \\
r_{3.1} &: \texttt{IF } E = 0 \texttt{ THEN } D = 1 \\
r_{3.2} &: \texttt{IF } F = 1 \texttt{ THEN } D = 1
\end{aligned}
$$

Zugehöriger Regelgraph G_R:



Bemerkungen:

- Die Zyklenfreiheit eines zusammenhängenden gerichteten Graphen kann in linearer Zeit ($O(E)$) festgestellt werden.
- Der Algorithmus BC-DFS ist korrekt für zyklenfreie Regelmengen.
- Die Voraussetzung „zyklenfrei“ ist notwendig, da Tiefensuche auf unendlichen Graphen keine vollständige Suchstrategie darstellt.

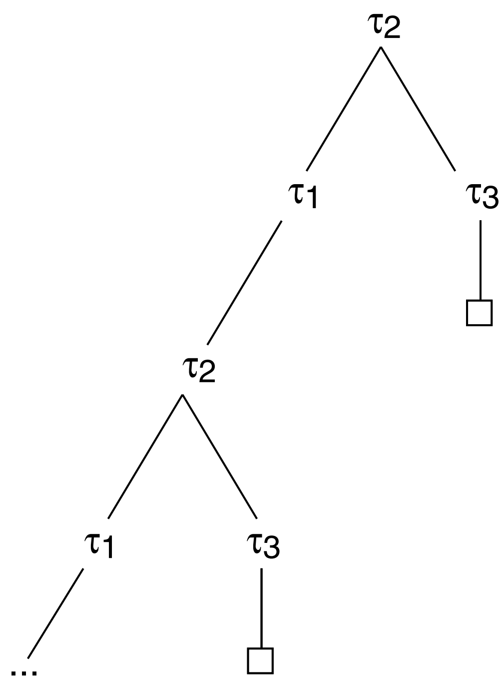
$\Rightarrow$ Im Zusammenhang mit rückwärtsverkettenden Verfahren und der Kontrollstrategie Tiefensuche ist es notwendig, Schleifen während der Abarbeitung zu erkennen.

Backward-Chaining

Beispiel zyklische Regelmenge. Sei folgendes Produktionsregelsystem $P = (D, R)$ gegeben:

$$D = \{\tau_3\} \qquad R = \{r_1 : \texttt{IF } \tau_1 \texttt{ THEN } \tau_2$$
$$r_2 : \texttt{IF } \tau_2 \texttt{ THEN } \tau_1$$
$$r_3 : \texttt{IF } \tau_3 \texttt{ THEN } \tau_2\}$$

Ableitungsbaum $T_P(\tau_2)$ für Ziel τ_2:



Backward-Chaining

Satz 2

Die Laufzeit des Algorithmus BC-DFS ist auch bei zyklenfreien Regelmengen R nicht durch ein Polynom in Abhängigkeit von der Größe von $P = (D, R)$ beschränkt.

Beweis 1

Gegeben sei folgendes Produktionsregelsystem $P_n = (D, R_n)$:

$$D = \{\tau_0\} \qquad R = \{\texttt{IF } \alpha_{i,1} \wedge \alpha_{i,2} \texttt{ THEN } \tau_i,\ \texttt{IF } \tau_{i-1} \texttt{ THEN } \alpha_{i,1},\ \texttt{IF } \tau_{i-1} \texttt{ THEN } \alpha_{i,2} \mid 1 \leq i \leq n\}$$

(die Anzahl der Regeln ist $3n$)

Sei $\mu(n)$ die Anzahl der Aufrufe von BC-DFS für das Ziel τ_n.

Backward-Chaining

Beweis Laufzeit BC-DFS (Fortsetzung).

- $n = 0$:

$$\mu(0) = 1, \text{ da } \tau_0 \in D.$$

- $n = 1$:

$\longrightarrow \texttt{BC-DFS}(\tau_1)$

$\quad \longrightarrow \texttt{BC-DFS}(\alpha_{1,1})$

$\quad\quad \longrightarrow \texttt{BC-DFS}(\tau_0)$ (⇒ Anzahl der Aufrufe für τ_0)

$\quad$ `AND`

$\quad \longrightarrow \texttt{BC-DFS}(\alpha_{1,2})$

$\quad\quad \longrightarrow \texttt{BC-DFS}(\tau_0)$ (⇒ Anzahl der Aufrufe für τ_0)

- Allgemein für $n > 0$:

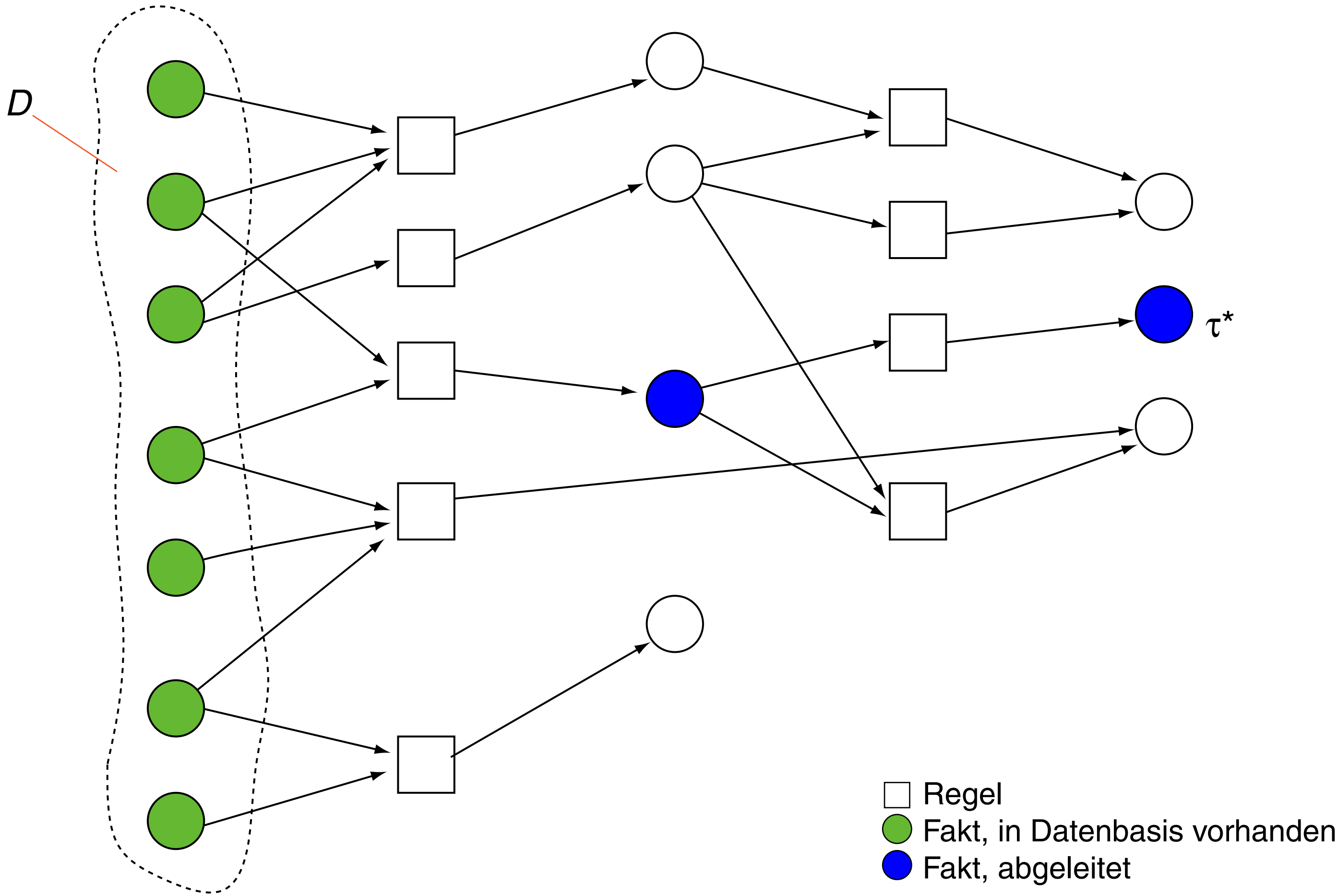
$$\mu(n) = 3 + 2 \cdot \mu(n-1) = 3 \cdot \sum_{i=0}^{n-1} 2^i + 2^{n+1} = 3\frac{1-2^n}{1-2} + 2^{n+1}$$

$$= 3 \cdot 2^n + 2^{n+1} - 3 \geq 2^n$$

Verkettungsstrategien

Frage: Ist ein Atom ableitbar?

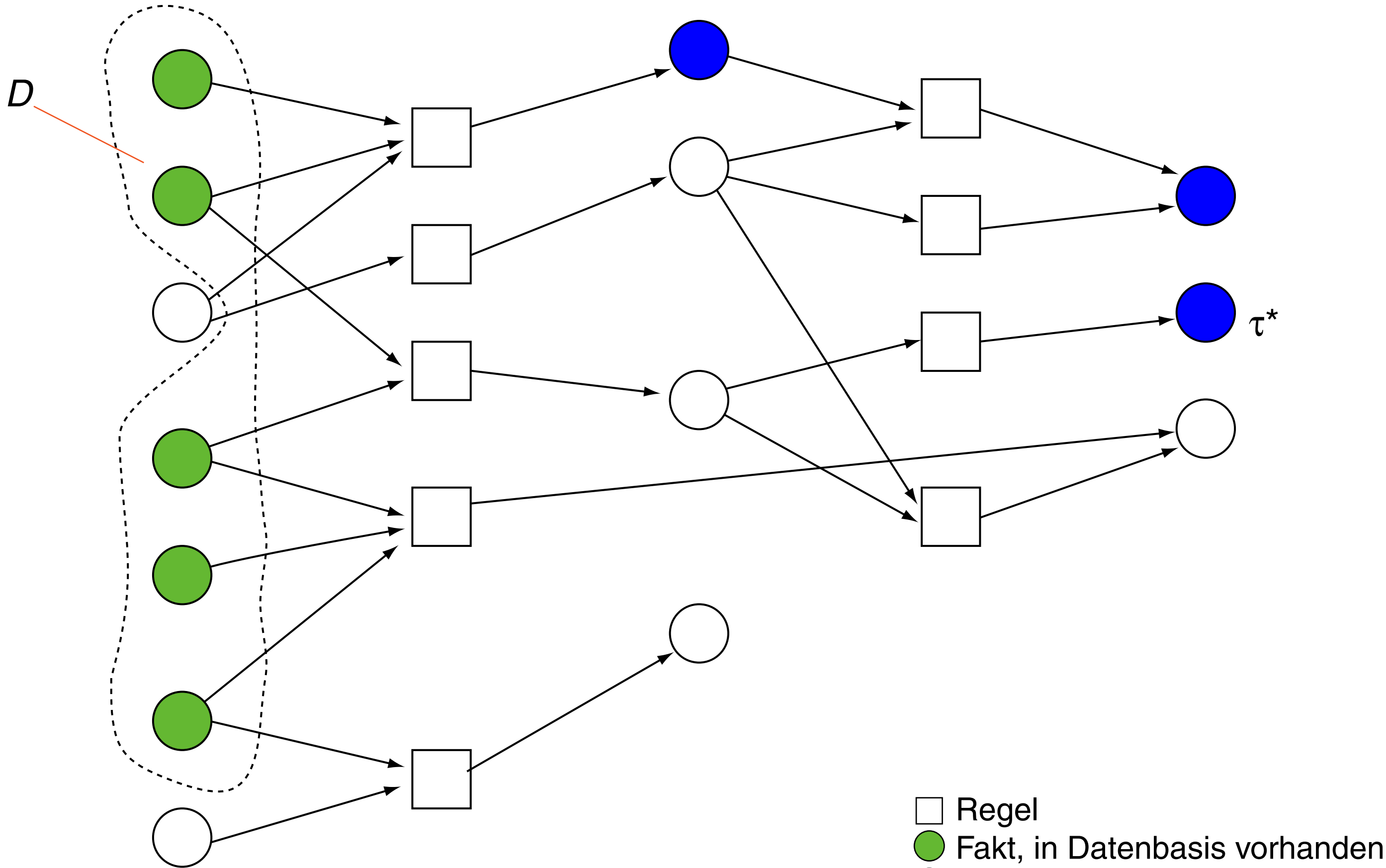
Bevorzugte Strategie: Backward-Chaining



Verkettungsstrategien

Frage: Wie sieht die Welt nach Anwendung aller Regeln aus?

Bevorzugte Strategie: Forward-Chaining



Verkettungsstrategien

Neben einer reinen Backward-Chaining oder Forward-Chaining-Strategie können auch Kombinationen hieraus sinnvoll sein: Inferenz rückwärts vom Ziel und „gleichzeitig“ vorwärts von den Fakten.

Gemischte Strategie:

1. *Fokussierung* durch Erzeugung einer Teilwelt D' durch Forward-Chaining.
2. Überprüfung von Hypothesen in D' mit Hilfe von Backward-Chaining.