Bachelor scriptie
Uitdrukkingskracht van XML schema's

Robbert Krebbers

24 juni 2008

Samenvatting

Er zijn vele verschillende XML schema's, waaronder DTD, XML schema en Relax NG, al deze schema's hebben verschillende eigenschappen. In deze scriptie zullen we een vergelijking maken wat betreft de uitdrukkingskracht van deze schema's. Hiervoor zullen we formele talen en automatentheorie gebruiken.

Inhoudsopgave

Dankwoord

Bij deze wil ik Herman Geuvers bedanken die mij attent maakte op het gebruik van formele talen en automatentheorie om te redeneren over de uitdrukkingskracht van XML schema's. Daarnaast wil ik hem bedanken voor zijn begeleiding bij het schrijven van deze scriptie.

Hoofdstuk 1

Inleiding

Momenteel bestaat het grootste gedeelte van het World Wide Web uit gegevens aangeboden in HTML formaat. In het HTML formaat is de betekenis van de elementen gedefinieerd als hun opmaak, voorbeelden hiervan zijn: een kop, een lijst, een tabel.

Met het semantische web, een evoluerende extensie van het World Wide Web is het de bedoeling betekenis te geven aan de data. Elementen representeren hierdoor objecten uit de werkelijkheid, zoals een persoon of een boek. Hierdoor kunnen de gegevens eenvoudiger door automatische processen worden verwerkt, in plaats dat ze enkel door een mens gelezen kunnen worden.

XML is een veelgebruikte techniek om gegevens op deze manier te representeren. Echter moet er op een bepaalde manier aangegeven worden hoe de structuur van gegevens in een XML document eruit ziet, anders definiëren verschillende mensen objecten op een andere manier waardoor er nog steeds niks mee gedaan kan worden.

Om aan te geven welke gegevens en met welke structuur je in een XML document kan uitdrukken zijn er zogenaamde XML schema's. Er zijn echter tientallen verschillende soorten XML schema's welke allemaal andere eigenschappen hebben. Naast dat de schema's er qua syntax op een andere manier uitzien is er ook een verschil in uitdrukkingskracht.

1.1 Probleemstelling

De onderzoeksvraag van mijn onderzoek luidt als volgt:

> Hoe verhouden de verschillende XML schema's zich qua uitdrukkingskracht wat betreft boomstructuren?

1.2 Methode

De onderzoeksvraag is op te splitsen in de volgende deelvragen:

1. Hoe kan een XML document formeel worden gedefinieerd?
2. Welke (noemenswaardige) XML schema's zijn er?

3. Welke formelen modellen zijn er om XML documenten te genereren en te accepteren?

4. Hoe verhouden de besproken modellen zich qua uitdrukkingskracht?

5. Hoe verhouden de besproken XML schema's zich tot de besproken modellen?

6. Wat is uiteindelijk te zeggen over de uitdrukkingskracht van de besproken XML schema's?

Zoals te lezen beperkt dit onderzoek zich tot de boomstructuur van een XML document, naar de volgende aspecten van XML en XML schema's wordt niet gekeken:

1. **Attributen**, naast dat er in een XML document een boom structuur van bepaalde elementen uit te drukken is, kunnen deze elementen ook attributen hebben. Een XML schema kan hier ook restricties op leggen.

2. **Gegevenstypes**, de gegevens die je in een XML document representeert kunnen bepaalde basistypes hebben, zoals een string of een integer. Een XML schema biedt enkele van deze basistypes aan en legt restricties wat betreft het gebruik van deze types op.

3. **Parseren**, een XML document wordt veelal als een string aangeboden welke vervolgens omgezet moet worden naar een boom.

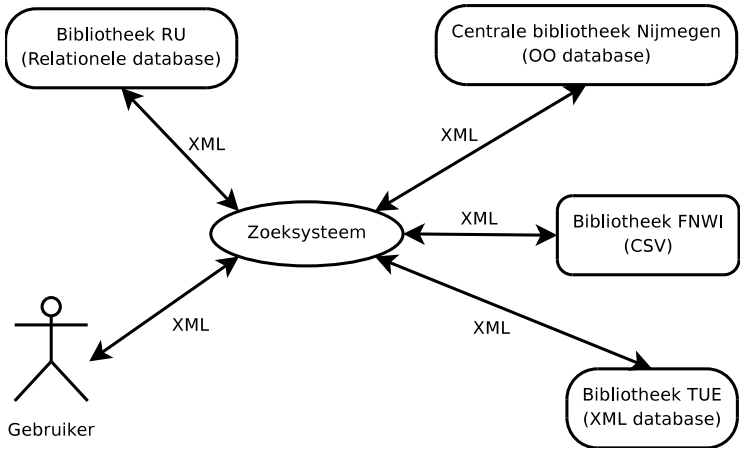
4. **Complexiteit**, het zal lastiger zijn om te controleren of een document aan het ene XML schema voldoet dan aan het andere.

De functie van het onderzoek is een vergelijking tussen enkele noemenswaardige XML schema's qua uitdrukkingskracht op te leveren.

Hoofdstuk 2

XML

Extensible Markup Language (XML) is een taal welke het mogelijk maakt gegevens tussen verschillende applicaties uit te wisselen, voornamelijk via het internet. Zie figuur 2.1 voor een fictieve situatie waarin verschillende soorten databases worden gebruikt om gegevens op te slaan. XML wordt gebruikt om gegevens tussen deze verschillende systemen en het centrale zoeksysteem uit te wisselen.



Figuur 2.1: XML als formaat om gegevens uit te wisselen

Het *extensible* komt doordat het mogelijk is als gebruiker zelf de te gebruiken elementen vast te leggen, dit gebeurt middels XML schema's, waarover in het volgende hoofdstuk meer.

De syntax van een XML document is als volgt gedefinieerd:

```
<?xml version="1.0" encoding="UTF-8"?>
element
```

`element` ziet er als volgt uit.

1. Een vrije tekst waarin speciale tekens ge-escaped zijn.

2. `<element-name>element</element-name>` waarbij `element` een element is. Dit wordt de vertakking genoemd.

3. `<element-name />`, dit is het lege element.

4. `element1 element2`, waarbij `element1` en `element2` elementen zijn. Dit is de concatenatie van 2 elementen.

Indien een XML document aan de regels hierboven beschreven voldoet, wordt dit XML document *well-formed* genoemd. Hieronder volgt een voorbeeld van een (well formed) XML document. Dit document bevat een rooster welke bestaat uit cursussen. Deze cursussen bestaan uit een titel, een aantal docenten en een aantal studenten.

```
<?xml version="1.0" encoding="UTF-8"?>
<course>
  <title>sl2</title>
  <person>
    <name>Herman</name>
    <department>fnds</department>
  </person>
  <person>
    <name>Robbert</name>
    <study>cs</name>
  </person>
</course>
```

2.1 Hedges

Een XML document is voor te stellen als een hedge, een gesorteerde ongenummerde rij van gesorteerde ongenummerde bomen.

Definitie 2.1.1. *Een* hedge *[3] over een eindige set van symbolen Σ en een eindige set variabelen X is als volgt gedefinieerd.*

1. *ϵ is een hedge, de null hedge.*

2. *x is een hedge, waarbij x een variabele in X is.*

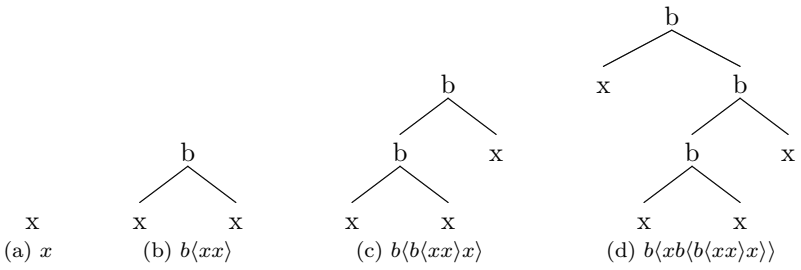
3. *$a\langle u\rangle$ is een hedge, waarbij a een symbool uit Σ is en u een hedge is. Dit wordt de vertakking genoemd.*

4. *uv is een hedge, waarbij u en v hedges zijn. Dit is de concatenatie van 2 hedges.*

Enkele hedges uit het alfabet $\Sigma = \{b\}$ en variabelen $X = \{x\}$ zien er als volgt uit: $b\langle xx\rangle$, $b\langle b\langle xx\rangle x\rangle$, $b\langle xb\langle b\langle xx\rangle x\rangle\rangle$. Zoals al snel te zien zijn het binaire bomen met symbool b als knoop en variabele x als blad. Zie figuur 2.2 voor een grafische weergave.

Definitie 2.1.2. *Een hedge is een* tree *indien de wortel een vertakking of een variabele is.*

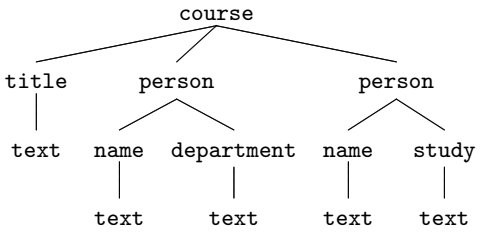
De bovengenoemde hedges zijn allemaal trees, de volgende hedges niet: $b\langle x\rangle b\langle x\rangle$, $xxxxx$, $b\langle xx\rangle xxx$, ϵ.



Figuur 2.2: Hedges van binaire bomen

2.2 Hedge van een XML document

In het geval van een XML document komt Σ overeen met de elementen in het XML document. X komt overeen met de daadwerkelijke data, of types van de daadwerkelijke data. Neem het XML document uit het begin van dit hoofdstuk als voorbeeld. Hierbij is $\Sigma = \{\texttt{course}, \texttt{title}, \texttt{person}, \texttt{name}, \texttt{department}, \texttt{study}\}$ en $X = \{\texttt{text}\}$. In figuur 2.3 is de hedge grafisch weergegeven.



Figuur 2.3: De hedge van een XML document

De hedge ϵ kan worden gebruikt om het lege document of lege elementen weer te geven. De hedge van `<br />` is $\texttt{br}\langle\epsilon\rangle$.

Hoofdstuk 3

XML schema's

Om het voor verschillende applicaties mogelijk te maken gegevens in XML formaat uit te wisselen, is het nodig de structuur van XML documenten te specificeren. Dezelfde informatie kan namelijk op verschillende manieren worden gerepresenteerd, neem de volgende XML documenten als voorbeeld:

```
<?xml version="1.0" encoding="UTF-8"?>
<persoon>
  <naam>
    <voor>Piet</voor>
    <achter>Puk</achter>
  </naam>
</persoon>
```

En:

```
<?xml version="1.0" encoding="UTF-8"?>
<persoon>
  <voornaam>Piet</voornaam>
  <achternaam>Puk</achternaam>
</persoon>
```

Daarom zijn zogenaamde XML schema's nodig, hiermee is het mogelijk de structuur van een XML document vast te leggen. XML schema's zijn grammatica's voor XML documenten.

Een XML document wordt *valid* genoemd indien die voldoet aan de regels beschreven in een bepaald XML schema.

In deze sectie worden enkele XML schema's beschreven. Merk op dat slechts een subset van deze schema's wordt beschreven, namelijk enkel het gedeelte dat relevant is wat betreft boomstructuur. Ook zijn er vaak vele notaties en constructies mogelijk welke allemaal resulteren in hetzelfde effect, er wordt slechts één notatie beschreven.

3.1 DTD

Een van de oudste XML schema's is *Document Type Definition (DTD)* [7], deze W3C standaard bestond zelfs al lang voor de tijd XML bestond, namelijk om SGML documenten te beschrijven. Een DTD ziet er als volgt uit:

```
<!DOCTYPE root-element [element-declarations]>
```

Waarbij `root-element` de wortel van het document aangeeft, en `element-declarations` een serie regels bevat welke er als volgt uitzien:

1. `<!ELEMENT name (content-model)>`

   Deze zijn op te vatten als productieregels waarbij `name` de naam van het element is en `content-model` een soort van reguliere expressie is, deze reguliere expressie is als volgt gedefinieerd.

   (a) Elk gedefinieerd element is een content-model.

   (b) `#PCDATA` is een content-model, dit zijn gegevens als string.

   (c) `x, y` is een content-model indien `x` en `y` een geldig content-model zijn. Dit is de concatenatie van `x` en `y`.

   (d) `x|y` is een content-model indien `x` en `y` een geldig content-model zijn. Dit is de keuze tussen uit `x` en `y`.

   (e) `x+` is een content-model indien `x` een geldig content-model is. Dit betekent dat `x` één of meerdere keren achter elkaar mag voorkomen.

   (f) `x*` is een content-model indien `x` een content-model is. Dit betekent dat `x` nul of meerdere keren achter elkaar mag voorkomen.

   (g) `x?` is een content-model indien `x` een content-model is. Dit betekent dat `x` nul of één keer mag voorkomen.

   (h) `(x)` is een content-model indien `x` een content-model is. Hiermee kan gegroepeerd worden.

   Een extra opmerkingen hierbij is dat de reguliere expressies deterministisch moeten zijn. Zie hiervoor sectie 3.1.1

2. `<!ELEMENT name EMPTY>`

   Waarbij `name` de naam van een element is, dit element behoort altijd leeg te zijn.

Er mogen geen meerdere `ELEMENT` regels voorkomen met dezelfde `name`. Merk op dat een DTD enkel trees genereert, gezien de wortel van een DTD een enkel element is.

De volgende DTD genereert binaire bomen:

```
<!DOCTYPE b [
  <!ELEMENT b ((b|x), (b|x))>
  <!ELEMENT x (#PCDATA)>
]>
```

Hier wordt een element `x` met `#PCDATA` als inhoud als blad gebruikt de bomen hebben daarnaast altijd een minimale diepte van één.

Roosters uit het voorbeeld in sectie 2.2 kunnen met de volgende DTD gegenereerd worden:

```
<!DOCTYPE schedule [
  <!ELEMENT schedule (course*)>
  <!ELEMENT course (title, person*)>
  <!ELEMENT person (name, (study+ | department*))>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT study (#PCDATA)>
  <!ELEMENT department (#PCDATA)>
]>
```

Merk al direct het extra element `schedule` bij de wortel op, dit komt omdat DTD's slechts bomen kunnen genereren.

3.1.1 Determinisme van de productieregels

Vanwege backwards compatibility met SGML moet het content model deterministisch zijn. Dit betekent dat de processor zonder vooruit te kijken de reguliere expressie toe kan passen. Het volgende content-model is non-deterministisch $(bc)|(bd)$, bij een b aan het begin weet de processor niet welke b hij moet kiezen, hij zal hiervoor vooruit moeten kijken. Bovenstaande is te herschrijven naar een deterministische equivalent: $b(c|d)$. Merk op dat niet elk non deterministisch content model om te schrijven is naar een deterministische equivalent, bijvoorbeeld $(a|c)^*a(a|c)$. Lees hierover meer in hoofdstuk 5.

3.2 XML schema

XML Schema [9], beschreven door de W3C is de opvolger van DTD. Een XML Schema wordt geschreven in XML syntax en ziet er als volgt uit (we gebruiken reguliere expressies als meta syntax):

```
<xs:schema>
  (complexType | element)*
</xs:schema>
```

Hierbij gelden de volgende definities:

1. `complexType`:

   Dit is een type definitie, deze bevat een reguliere expressie over elementen. Bij een element kan aangegeven worden tot wat voor een type deze behoord.

   ```
   <xs:complexType name="type">
     (choice | sequence)
   </xs:complexType>
   ```

   Waarbij:

   - `type` een unieke type naam is, deze identificeert dit type.
   - Met behulp van `choice` en `sequence` wordt in XML notatie een soort van reguliere expressie opgeschreven.

2. `sequence`:

   Deze geeft aan dat de kinderen in opgegeven volgorde moeten voorkomen.

   ```
   <xs:sequence minOccurs="min" maxOccurs="max">
     (element | choice | sequence)*
   </xs:sequence>
   ```

   Waarbij:

   - `min` een positieve integer is. Deze geeft het minimale aantal keer dat deze rij achter elkaar mag voorkomen.
   - `max` een positieve integer of `unbounded` is. Deze geeft het maximale aantal keer dat deze rij achter elkaar mag voorkomen.

3. `choice`:

   Deze geeft aan dat één van de kinderen moet voorkomen.

   ```
   <xs:choice minOccurs="min" maxOccurs="max">
     (element | choice | sequence)*
   </xs:choice>
   ```

   Waarbij:

   - `min` een positieve integer is. Deze geeft het minimale aantal keer dat deze optie achter elkaar mag voorkomen.
   - `max` een positieve integer of `unbounded` is. Deze geeft het maximale aantal keer dat deze optie achter elkaar mag voorkomen.

4. `element`:

   Een element wordt *lokaal* genoemd indien deze een `xs:complexType` als voorouder heeft. Een element wordt *globaal* genoemd indien deze `xs:schema` als directe ouder heeft.

   (a) `<xs:element name="name" type="type" minOccurs="min" maxOccurs="max" />`

   Waarbij:

   - `name` een element naam is.
   - `type` een type naam van een gedefinieerd `xs:complexType` type is of `xs:string` is.

   De volgende eigenschappen zijn enkel van toepassing als het element *lokaal* is:

   - `min` een positieve integer is. Deze geeft het minimale aantal keer dat het element achter elkaar mag voorkomen.
   - `max` een positieve integer of `unbounded` is. Deze geeft het maximale aantal keer dat het element achter elkaar mag voorkomen.

   (b)
   ```
   <xs:element name="name" minOccurs="min" maxOccurs="max">
     complexType
   </xs:element>
   ```

   Waarbij:

   - `name` een element naam is.

   De volgende eigenschappen zijn enkel van toepassing als het element *lokaal* is:

   - `min` een positieve integer is. Deze geeft het minimale aantal keer dat het element achter elkaar mag voorkomen.
   - `max` een positieve integer of `unbounded` is. Deze geeft het maximale aantal keer dat het element achter elkaar mag voorkomen.

Merk op dat er bij XML Schema niet expliciet een start aangegeven wordt, elk *globaal* element mag als start element worden gebruikt.

Het volgende schema genereert binaire bomen, ook deze gebruikt het element `x` met `xs:string` als blad.

```
<xs:schema>
  <xs:complexType name="b">
    <xs:choice minOccurs="2" maxOccurs="2">
      <element name="b" type="b" />
      <element name="x" type="xs:string" />
    <xs:choice>
  </xs:complexType>

  <xs:element name="b" type="b" />
  <xs:element name="x" type="xs:string" />
</xs:schema>
```

Roosters uit het voorbeeld in sectie 2.2 kunnen met het volgende XML schema gegenereerd worden:

```
<xs:schema>
  <xs:element name="schedule"><xs:complexType><xs:sequence>
    <xs:element name="course" type="course" minOccurs="1" maxOccurs="
        unbounded" />
  </xs:sequence><xs:complexType></xs:element>

  <xs:complexType name="course"><xs:sequence>
    <xs:element name="title" type="xs:string" />
    <xs:element name="person" type="person" minOccurs="1" maxOccurs="
        unbounded" />
  </xs:sequence><xs:complexType>

  <xs:complexType name="person"><xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:choice>
      <xs:element name="study" type="xs:string" minOccurs="1" maxOccurs="
          unbounded" />
      <xs:element name="department" type="xs:string" minOccurs="0" maxOccurs=
          "unbounded" />
    </xs:choice>
  </xs:sequence><xs:complexType>
</xs:schema>
```

Merk al direct het extra element `schedule` bij de wortel op, dit komt omdat XML schema's slechts bomen kunnen genereren.

3.2.1 Element Declarations Consistent

Volgens de Element Declarations Consistent (EDC) mogen binnen elk `complexType` geen elementen met dezelfde naam maar met verschillende types voorkomen. Het volgende (deel van een) XML Schema is bijvoorbeeld niet toegestaan:

```
<xs:element name="course">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="text" minOccurs="1" maxOccurs="1" />
      <xs:element name="person" type="teacher" minOccurs="1" maxOccurs="
          unbounded" />
      <xs:element name="person" type="student" minOccurs="1" maxOccurs="
          unbounded" />
    </xs:sequence>
  <xs:complexType>
</xs:element>
```

`person` komt hier namelijk met zowel het type `teacher` en `student` voor in één `complexType`.

3.2.2 Unique Particle Attribution

XML-schema bevat een contraint welke zegt dat de processor zonder vooruit kijken de reguliere expressie binnen een `complexType` toe kan passen. In andere

woorden, ook de reguliere expressies welke in een XML schema gebruikt mogen worden moeten deterministisch zijn.

3.3 Relax NG

REgular LAnguage for XML Next Generation (RELAX NG) [6] is een XML schema gebaseerd op Murata Makoto's RELAX en James Clark's TREX.

Er zijn 2 notaties om een Relax NG schema op te schrijven, namelijk in XML syntax en als verkorte notatie. We zullen hier enkel de verkorte notatie behandelen en een voorbeeld geven van een schema in de XML notatie. Een RelaxNG schema ziet er als volgt uit:

```
grammar {
  start = expression
  non-terminals
}
```

Waarbij `non-terminals` uit nul of meer van de volgende regels bestaat:

```
non-terminal = expression
```

`non-terminal` is een string, welke niet gelijk is aan een van de reserveerde woorden: `element`, `attribute`, `text`, `empty` en `grammar`. `expression` is hierbij als volgt gedefinieerd:

1. `non-terminal` is een expressie indien `non-terminal` een non-terminal is en niet gelijk is aan de non-terminal van de directe vader.
2. `empty` is een expressie.
3. `text` is een expressie. Dit zijn gegevens als string.
4. `element elementname { expression }` is een expressie indien `expression` een expressie is. Dit geeft een vertakking in de boom aan.
5. `x, y` is een expressie indien `x` en `y` expressies zijn. Dit is de concatenatie van `x` en `y`.
6. `x+` is een expressie, indien `x` een expressie is. Dit betekend dat `x` één of meerdere keren achter elkaar mag voorkomen.
7. `x*` is een expressie, indien `x` een expressie is. Dit betekend dat `x` nul of meerdere keren achter elkaar mag voorkomen.
8. `x?` is een expressie, indien `x` een expressie is. Dit betekend dat `x` nul of één keer mag voorkomen.
9. `(x)` is een expressie, indien `x` een expressie is. Hiermee kan gegroepeerd worden.

Voor het voorbeeld van de binaire bomen uit 2.1 is het volgende RelaxNG schema te geven:

```
grammar {
  start = b
  b = (x | element b {b}), (x | element b {b})
  x = element x { text }
}
```

Roosters uit het voorbeeld in sectie 2.2 kunnen met de volgende RelaxNG gegenereerd worden:

```
grammar {
  start = element course { element title { text }, teacher+, student+ }+
  teacher = element person { name, department* }
  student = element person { name, study+ }
  name = element name { text }
  study = element study { text }
  department = element department { text }
}
```

In XML notatie ziet dit er als volgt uit:

```
<grammar>
  <start><element name="course">
    <ref name="title"/>
    <oneOrMore><ref name="teacher"/></oneOrMore>
    <oneOrMore><ref name="student"/></oneOrMore>
  </element></start>

  <element name="teacher">
    <ref name="name"/>
    <zeroOrMore><ref name="department"/></zeroOrMore>
  </element>

  <element name="student">
    <ref name="name"/>
    <oneOrMore><ref name="study"/></oneOrMore>
  </element>

  <element name="title"><text/></element>
  <element name="name"><text/></element>
  <element name="department"><text/></element>
  <element name="study"><text/></element>
</grammar>
```

Of eventueel wat compacter geschreven:

```
grammar {
  start = element course {
    element title { text },
    element person { element name { text }, element department { text }* }+,
    element person { element name { text }, element study { text }+ }+
  }+
}
```

In XML notatie ziet dit er als volgt uit:

```
<grammar>
  <start>
    <element name="course">
      <element name="title"><text/></element>
      <oneOrMore>
        <element name="name"><text/></element>
        <zeroOrMore><element name="department"><text/></element></zeroOrMore>
      </oneOrMore>
      <oneOrMore>
        <element name="name"><text/></element>
        <oneOrMore><element name="study"><text/></element></oneOrMore>
      </oneOrMore>
    </element>
  </start>
</grammar>
```

Hoofdstuk 4

Hedge grammatica's en automaten

In deze sectie geven we enkel formelen modellen welke het mogelijk maken om bepaalde hedges te genereren of te accepteren. Deze zullen grote overeenkomsten vertonen met de besproken XML schema's. Deze modellen zullen de uiteindelijk vergelijking tussen de XML schema's mogelijk maken.

4.1 Hedge grammatica's

In deze sectie introduceren we reguliere hedge grammatica's, ook wel RHG's genoemd. Deze maken het mogelijk om hedges te genereren.

Definitie 4.1.1. *Een* reguliere hedge grammatica (RHG) *[3] is een 5 tupel,* $\langle \Sigma, X, N, P, r_f \rangle$*, waarbij:*

1. Σ *een eindige set van symbolen is.*
2. X *een eindige set van variabelen is.*
3. N *een eindige set van non-terminals is.*
4. P *een eindige set van productie regels is, deze zien er als volgt uit:*
   (a) $n \rightarrow x$*, waarbij* n *een non-terminal uit* N *is en* x *een variabele uit* X *is.*
   (b) $n \rightarrow a\langle r \rangle$ *waarbij* a *een symbool uit* Σ *en* r *een reguliere expressie gebruikmakend van de non-terminals,* N*, is.*
5. r_f *een reguliere expressie gebruikmakend van de non-terminals,* N*, is. Deze geeft de wortel van de hedge aan.*

Definitie 4.1.2. $n \twoheadrightarrow u$ *betekent dat de tree* u *in 0 of meerdere stappen wordt gegenereerd uit de non-terminal* n*.*

Definitie 4.1.3. $r \twoheadrightarrow u$ *betekent dat de hedge* u *in 0 of meerdere stappen wordt gegenereerd uit de reguliere expressie* r*.*

Definitie 4.1.4. *De taal $L(G)$ van een RHG G is de set van hedges welke de RHG genereert.*

$$L(G) = \{u \mid r_f \twoheadrightarrow u\}$$

Neem het voorbeeld van de binaire bomen uit sectie 2.1, een RHG, $G = \langle \Sigma, X, N, P, r_f \rangle$, welke binaire bomen genereert ziet er als volgt uit:

$$\begin{aligned} \Sigma &= \{b\} \\ X &= \{x\} \\ N &= \{n_b, n_x\} \\ P &= \{n_b \to b\langle (n_b|n_x)(n_b|n_x)\rangle, \\ &\quad\ n_x \to x\} \\ r_f &= n_b|n_x \end{aligned}$$

Merk op dat de volgende RHG, $G_2 = \langle \Sigma, X, N, P, r_f \rangle$, equivalent is:

$$\begin{aligned} \Sigma &= \{b\} \\ X &= \{x\} \\ N &= \{n\} \\ P &= \{n \to b\langle nn\rangle, \\ &\quad\ n \to x\} \\ r_f &= n \end{aligned}$$

Met behulp van een RHG is het dus eenvoudig mogelijk om alle mogelijke hedges, ook wel de taal van de RHG, $L(G)$, te genereren. In het geval van bovenstaande voorbeeld:

$$L(G) = \{x, b\langle xx\rangle, b\langle b\langle xx\rangle x\rangle, b\langle xb\langle xx\rangle\rangle, b\langle b\langle xx\rangle b\langle xx\rangle\rangle, b\langle b\langle b\langle xx\rangle x\rangle b\langle xx\rangle\rangle, \ldots\}$$

4.1.1 RHG voor een XML document

Voor het voorbeeld van het rooster uit sectie 2.2 is de volgende RHG, $G = \langle \Sigma, X, N, P, r_f \rangle$ te geven, waarbij:

$$\begin{aligned} \Sigma &= \{\texttt{course}, \texttt{title}, \texttt{person}, \texttt{name}, \texttt{study}, \texttt{department}\} \\ X &= \{\texttt{text}\} \\ N &= \{n_{course}, n_{teacher}, n_{student}, n_{title}, n_{name}, n_{department}, n_{study}, n_{text}\} \\ P &= \left\{ \begin{array}{rcl} n_{course} & \to & \texttt{course}\langle n_{title} {n_{teacher}}^{+} {n_{student}}^{+}\rangle, \\ n_{teacher} & \to & \texttt{person}\langle n_{name} {n_{department}}^{*}\rangle, \\ n_{student} & \to & \texttt{person}\langle n_{name} {n_{study}}^{+}\rangle, \\ n_{title} & \to & \texttt{title}\langle n_{text}\rangle, \\ n_{name} & \to & \texttt{name}\langle n_{text}\rangle, \\ n_{department} & \to & \texttt{department}\langle n_{text}\rangle, \\ n_{study} & \to & \texttt{study}\langle n_{text}\rangle, \\ n_{text} & \to & \texttt{text} \end{array} \right\} \\ r_f &= {n_{course}}^{+} \end{aligned}$$

Deze RHG zorgt er voor dat elke cursus een datum, minstens één docent en één student heeft. Een docent heeft een naam en werkt op nul of meerdere afdelingen. Een student heeft een naam en doet één of meerdere studies.

4.2 Hedge automaten

4.2.1 Deterministische hedge automaten

Om te controleren of een hedge aan een bepaalde grammatica voldoet introduceren we deterministische hedge automaten, ook wel DHA's genoemd.

Definitie 4.2.1. *Een* deterministische hedge automaat (DHA) *[3] is een 6 tupel,* $\langle \Sigma, X, Q, \alpha, \iota, F \rangle$*, waarbij:*

1. Σ *een eindige set van symbolen is.*
2. X *een eindige set van variabelen is.*
3. Q *een eindige set van states is.*
4. α *een functie van* $\Sigma \times Q^*$ *naar* Q *is. Hierbij geldt dat dat voor elke* $q \in Q$ *en* $z \in \Sigma$ *dat* $\{\bar{q} \mid \alpha(z, \bar{q}) = q\}$ *een reguliere set is.*
5. ι *een functie van* X *naar* Q *is.*
6. F *een reguliere set van geaccepteerde eindstates,* 2^{Q^*}*, is.*

Om te controleren of een hedge door een DHA geaccepteerd wordt ga je bottom up te werk.

Definitie 4.2.2. *Een hedge* u *wordt geaccepteerd [4] indien* $M\|u \in F$*. Waarbij* $M\|u$ *recursief als volgt gedefinieerd is:*

$$\begin{aligned} M\|\epsilon &= \epsilon \\ M\|x &= \iota(x) \\ M\|a\langle u \rangle &= \alpha(a, M\|u) \\ M\|uv &= (M\|u)(M\|v) \end{aligned}$$

Een DHA levert een rij met toestanden op, dus: $M\|u \in Q^*$*.*

Definitie 4.2.3. *De taal* $L(M)$ *van een DHA* M *is de set van hedges welke de DHA accepteert.*

Om overzichtelijk de controle van een hedge op te schrijven maken we er een afleidingsboom van. Er zijn 4 bewijsstappen, welke corresponderen met de bovenstaande formules. 2 er van zijn axioma's, deze zijn als volgt:

$$\frac{}{M\|\epsilon = \epsilon} \text{ null}$$

$$\frac{}{M\|x = \iota(x)} \text{ variabele}$$

Daarnaast zijn er de volgende overige bewijsstappen:

$$\frac{M\|u = \bar{q}}{M\|a\langle u \rangle = \alpha(a, \bar{q})} \text{ vertakking}$$

$$\frac{M\|u = \bar{q_u} \qquad M\|v = \bar{q_v}}{M\|uv = \bar{q_u}\bar{q_v}} \text{ concatenatie}$$

Neem het voorbeeld van de binaire bomen uit sectie 2.1, een DHA, $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, welke binaire bomen accepteert ziet er als volgt uit:

$$\begin{aligned} \Sigma &= \{b\} \\ X &= \{x\} \\ Q &= \{q_b, q_x, q_0\} \\ \iota(x) &= q_x \\ \alpha(b, \bar{q}) &= \begin{cases} q_b & \text{als } \bar{q} \in L((q_b|q_x)(q_b|q_x)) \\ q_0 & \text{anders} \end{cases} \\ F &= L(q_b|q_x) \end{aligned}$$

Het controleren van de hedge $b\langle b\langle xx\rangle x\rangle$ gaat als volgt:

1. Bereken $M\|b\langle b\langle xx\rangle x\rangle$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{M\|x = q_x} \quad \overline{M\|x = q_x}}{M\|xx = q_x q_x}}{M\|b\langle xx\rangle = q_b} \quad \overline{M\|x = q_x}}{M\|b\langle xx\rangle x = q_b q_x}}{M\|b\langle b\langle xx\rangle x\rangle = q_b}$$

2. Controleer of $M\|b\langle b\langle xx\rangle x\rangle = q_b \in F = L(q_b)$.
3. Dit klopt, dus $b\langle b\langle xx\rangle x\rangle \in L(M)$ ✓.

Echter kan het natuurlijk ook voorkomen dat een automaat een verkeerde hedge ter controle krijgt. We laten bovenstaande automaat nu de volgende hedge controleren: $b\langle b\langle xxx\rangle xx\rangle$ (een ternaire boom). Dit gaat als volgt:

1. Bereken $M\|b\langle b\langle xxx\rangle xx\rangle$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{M\|x = q_x} \quad \overline{M\|x = q_x}}{M\|xx = q_x q_x} \quad \overline{M\|x = q_x}}{M\|xxx = q_x q_x q_x}}{M\|b\langle xxx\rangle = q_0} \quad \overline{M\|x = q_x}}{M\|b\langle xxx\rangle x = q_0} \quad \overline{M\|x = q_x}}{M\|b\langle xxx\rangle xx = q_0 q_x q_x}}{M\|b\langle b\langle xxx\rangle xx\rangle = q_0}$$

2. Controleer of $M\|b\langle b\langle xxx\rangle xx\rangle = q_0 \in F = L(q_b)$.
3. Dit klopt niet, dus $b\langle b\langle xxx\rangle xx\rangle \notin L(M)$.

4.2.2 DHA voor een XML document

Voor het voorbeeld van het rooster uit sectie 2.2 is de volgende DHA te geven $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, waarbij:

$$
\begin{aligned}
\Sigma &= \{\texttt{course}, \texttt{title}, \texttt{person}, \texttt{name}, \texttt{study}, \texttt{department}\} \\
X &= \{\texttt{text}\} \\
Q &= \{q_t, q_c, q_{ti}, q_{te}, q_{st}, q_n, q_s, q_d\} \\
\iota(\texttt{text}) &= q_t \\
\alpha(\texttt{course}, \bar{q}) &= \begin{cases} q_c & \text{als } \bar{q} \in L(q_{ti} q_{te}^{+} q_{st}^{+}) \\ q_0 & \text{anders} \end{cases} \\
\alpha(\texttt{person}, \bar{q}) &= \begin{cases} q_{te} & \text{als } \bar{q} \in L(q_n q_d^{*}) \\ q_{st} & \text{als } \bar{q} \in L(q_n q_s^{+}) \\ q_0 & \text{anders} \end{cases} \\
\alpha(\texttt{title}, \bar{q}) &= \begin{cases} q_{ti} & \text{als } \bar{q} \in L(q_t) \\ q_0 & \text{anders} \end{cases} \\
\alpha(\texttt{name}, \bar{q}) &= \begin{cases} q_n & \text{als } \bar{q} \in L(q_t) \\ q_0 & \text{anders} \end{cases} \\
\alpha(\texttt{study}, \bar{q}) &= \begin{cases} q_s & \text{als } \bar{q} \in L(q_t) \\ q_0 & \text{anders} \end{cases} \\
\alpha(\texttt{department}, \bar{q}) &= \begin{cases} q_d & \text{als } \bar{q} \in L(q_t) \\ q_0 & \text{anders} \end{cases} \\
F &= L(q_c^{+})
\end{aligned}
$$

Hieronder is te zien hoe het rooster uit sectie 2.1 wordt gecontroleerd.

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{M \| \texttt{text} = q_t}}{M \| \texttt{title}\langle\texttt{text}\rangle = q_{ti}} \quad \dfrac{\dfrac{\dfrac{}{M \| \texttt{text} = q_t}}{M \| \texttt{name}\langle\texttt{text}\rangle = q_n} \quad \dfrac{\dfrac{}{M \| \texttt{text} = q_t}}{M \| \texttt{department}\langle\texttt{text}\rangle = q_d}}{M \| \texttt{name}\langle\texttt{text}\rangle\texttt{department}\langle\texttt{text}\rangle = q_n\, q_d}}{M \| \texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{department}\langle\texttt{text}\rangle\rangle = q_{te}} \quad \dfrac{\dfrac{\dfrac{}{M \| \texttt{text} = q_t}}{M \| \texttt{name}\langle\texttt{text}\rangle = q_n} \quad \dfrac{\dfrac{}{M \| \texttt{text} = q_t}}{M \| \texttt{study}\langle\texttt{text}\rangle = q_s}}{M \| \texttt{name}\langle\texttt{text}\rangle\texttt{study}\langle\texttt{text}\rangle = q_n\, q_s}}{M \| \texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{study}\langle\texttt{text}\rangle\rangle = q_{st}}}{M \| \texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{department}\langle\texttt{text}\rangle\rangle\texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{study}\langle\texttt{text}\rangle\rangle = q_{te}\, q_{st}}}{M \| \texttt{title}\langle\texttt{text}\rangle\texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{department}\langle\texttt{text}\rangle\rangle\texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{study}\langle\texttt{text}\rangle\rangle\rangle = q_{ti}\, q_{te}\, q_{st}}}{M \| \texttt{course}\langle\texttt{title}\langle\texttt{text}\rangle\texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{department}\langle\texttt{text}\rangle\rangle\texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{study}\langle\texttt{text}\rangle\rangle\rangle\rangle = q_c}
$$

4.2.3 Non-deterministische hedge automaten

In deze sectie introduceren we non-deterministische hedge automaten, ook wel NDHA's genoemd.

Definitie 4.2.4. *Een* non-deterministische hedge automaat (NDHA) *[3] is een 6 tupel,* $\langle \Sigma, X, Q, \alpha, \iota, F \rangle$*, waarbij:*

1. Σ *een eindige set van symbolen is.*
2. X *een eindige set van variabelen is.*
3. Q *een eindige set van states is.*

4. α *een functie van* $\Sigma \times Q^*$ *naar* 2^Q *is. Hierbij geldt dat dat voor elke* $q \in Q$ *en* $z \in \Sigma$ *dat* $\{\bar{q} \mid q \in \alpha(z, \bar{q})\}$ *een reguliere set is.*

5. ι *een functie van* X *naar* 2^Q *is.*

6. F *een reguliere set van geaccepteerde eindstates,* 2^{Q^*}*, is.*

Om te controleren of een hedge door een NDHA geaccepteerd wordt ga je bottom up te werk.

Definitie 4.2.5. *Een hedge* u *wordt geaccepteerd [4] indien* $(M\|u) \cap F$ *niet leeg is. Waarbij* $M\|u$ *recursief als volgt gedefinieerd is:*

$$\begin{aligned} M\|\epsilon &= \{\epsilon\} \\ M\|x &= \iota(x) \\ M\|a\langle u\rangle &= \{q \mid q \in \alpha(a, \bar{q}), \bar{q} \in M\|u\} \\ M\|uv &= \{\bar{q_u}\bar{q_v} \mid \bar{q_u} \in M\|u, \bar{q_v} \in M\|v\} \end{aligned}$$

Een NDHA levert een set van rijen met toestanden op, dus: $M\|u \in 2^{Q^*}$.

Definitie 4.2.6. *De taal* $L(M)$ *van een NDHA* M *is de set van hedges welke de NDHA accepteert.*

Om overzichtelijk de controle van een hedge op te schrijven maken we er een afleidingsboom van. Er zijn 4 bewijsstappen, welke corresponderen met de bovenstaande formules. 2 er van zijn axioma's, deze zijn als volgt:

$$\frac{}{M\|\epsilon = \{\epsilon\}}\ \text{null}$$

$$\frac{}{M\|x = \iota(x)}\ \text{variabele}$$

Daarnaast zijn er de volgende overige bewijstappen:

$$\frac{M\|u = X}{M\|a\langle u\rangle = \{q \mid q \in \alpha(a, \bar{q}), \bar{q} \in X\}}\ \text{vertakking}$$

$$\frac{M\|u = X_u \qquad M\|v = X_v}{M\|uv = \{\bar{q_u}\bar{q_v} \mid \bar{q_u} \in X_u, \bar{q_v} \in X_v\}}\ \text{concatenatie}$$

Stel we willen een automaat maken welke enkel binaire bomen met een maximale diepte van 3 toestaat. De RHG, $G = \langle \Sigma, X, N, P, r_f \rangle$, is vrij eenvoudig:

$$\begin{aligned} \Sigma &= \{b\} \\ X &= \{x\} \\ N &= \{n_{b1}, n_{b2}, n_{b3}, n_x\} \\ P &= \begin{Bmatrix} n_{b1} & \to & b\langle (n_{b2}|n_x)(n_{b2}|n_x)\rangle, \\ n_{b2} & \to & b\langle (n_{b3}|n_x)(n_{b3}|n_x)\rangle, \\ n_{b3} & \to & b\langle n_x n_x\rangle, \\ n_x & \to & x \end{Bmatrix} \\ r_f &= n_{b1}|n_x \end{aligned}$$

In tegenstelling tot een DHA is hier op leesbare wijze een NDHA voor op te schrijven, $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$:

$$\begin{array}{rcll} \Sigma & = & \{b\} & \\ X & = & \{x\} & \\ Q & = & \{q_{b1}, q_{b2}, q_{b3}, q_x\} & \\ \iota(x) & = & \{q_x\} & \\ \alpha(b, \bar{q}) & \ni & q_{b1} & \text{als } \bar{q} \in L((q_{b2}|q_x)(q_{b2}|q_x)) \\ \alpha(b, \bar{q}) & \ni & q_{b2} & \text{als } \bar{q} \in L((q_{b3}|q_x)(q_{b3}|q_x)) \\ \alpha(b, \bar{q}) & \ni & q_{b3} & \text{als } \bar{q} \in L(q_x q_x) \\ F & = & L(q_{b1}|q_x) & \end{array}$$

Het controleren van de hedge $b\langle b\langle b\langle xx\rangle x\rangle x\rangle$ gaat als volgt:

1. Bereken $M\|b\langle b\langle b\langle xx\rangle x\rangle x\rangle$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M\|x = \{q_x\}} \quad \dfrac{}{M\|x = \{q_x\}}}{M\|xx = \{q_x q_x\}}}{M\|b\langle xx\rangle = \{q_{b1}, q_{b2}, q_{b3}\}} \quad \dfrac{}{M\|x = \{q_x\}}}{M\|b\langle xx\rangle x = \{q_{b1}q_x, q_{b2}q_x, q_{b3}q_x\}}}{M\|b\langle b\langle xx\rangle x\rangle = \{q_{b1}, q_{b2}\}} \quad \dfrac{}{M\|x = \{q_x\}}}{M\|b\langle b\langle xx\rangle x\rangle x = \{q_{b1}q_x, q_{b2}q_x\}}}{M\|b\langle b\langle b\langle xx\rangle x\rangle x\rangle = \{q_{b1}\}}$$

2. Controleer of $M\|b\langle b\langle b\langle xx\rangle x\rangle x\rangle \cap F$, oftwel $\{q_b\} \cap L(q_b|q_x)$, niet leeg is.

3. Dit klopt, dus $b\langle b\langle b\langle xx\rangle x\rangle x\rangle \in L(M)$ ✓.

Zoals te zien is het met een NDHA makkelijk om een automaat op te schrijven waarbij een bepaald symbool uit Σ afhankelijk van zijn plek andere kinderen mag hebben (in bovenstaand voorbeeld; de knoop mag afhankelijk van zijn plek geen knopen meer onder zich hebben).

4.3 Equivalentie

In deze sectie zullen we laten zien dat de klasse talen gegenereerd door RHG's en geaccepteerd door (N)DHA's equivalent zijn. Dit is ook te lezen in [3], echter ontbreken constructies om de verschillende conversies te verwezenlijken en bewijzen in dat artikel.

Om enkele bewijzen in deze sectie wat makkelijker te maken zullen we bij DHA's de volgende extra bewijs stappen toestaan:

$$\frac{M\|u_1 = \bar{q_1} \qquad \cdots \qquad M\|u_n = \bar{q_n}}{M\|u_1 \ldots u_n = \bar{q_1} \ldots \bar{q_n}} \text{ concatenatie2}$$

$$\frac{M\|u_1 = \bar{q_1} \qquad \cdots \qquad M\|u_n = \bar{q_n}}{M\|a\langle u_1 \ldots u_n\rangle = \alpha(a, \bar{q_1} \ldots \bar{q_1})} \text{ vertakking2}$$

Met behulp van de 'concatenatie' en de 'null' stap is met inductie eenvoudig te zien dat de 'concatenatie2' stap correct is, met behulp van de 'concatenatie2' en de 'vertakking' stap is eenvoudig te zien dat de 'vertakking2' stap correct is.

Bij NDHA's voegen we gelijknamige regels toe, namelijk:

$$\frac{M\|u_1 = X_1 \qquad \cdots \qquad M\|u_n = X_n}{M\|u_1 \ldots u_n = \{q_1 \ldots q_n \mid q_1 \in X_1, \ldots, q_n \in X_n\}} \text{ concatenatie2}$$

$$\frac{M\|u_1 = X_1 \qquad \cdots \qquad M\|u_n = X_n}{M\|a\langle u_1 \ldots u_n\rangle = \{q \mid q \in \alpha(a, q_1 \ldots q_n), q_1 \in X_1, \ldots, q_n \in X_n\}} \text{ vertakking2}$$

4.3.1 RHG ⇒ NDHA

Stelling 4.3.1. *Bij iedere RHG G kan een NDHA M gemaakt worden zodat $L(G) = L(M)$.*

Bewijs. Gegeven een RHG $G = \langle \Sigma, X, N, P, r_f\rangle$, maak een NDHA $M = \langle \Sigma', X', Q', \alpha', \iota', F'\rangle$, waarbij:

$$\begin{aligned}
\Sigma' &= \Sigma \\
X' &= X \\
Q' &= N \\
n \in \iota'(x) &\leftrightarrow (n \to x) \in P \\
n \in \alpha'(a, \bar{q}) &\leftrightarrow \bar{q} \in L(r) \wedge (n \to a\langle r\rangle) \in P \\
F' &= L(r_f)
\end{aligned}$$

Om te bewijzen dat $L(G) = L(M)$ bewijzen we eerst het volgende lemma: $\boxed{n \twoheadrightarrow u \text{ desda } n \in M\|u}$ voor elke tree u en non-terminal n (ofwel state, gezien $Q' = N$). Dit gaat met behulp van inductie.

Basis:

$$\begin{aligned}
& n \to x && \text{(a)} \\
& \text{desda } n \in \iota'(x) && \text{(b)} \\
& \text{desda } n \in M\|x && \text{(c)}
\end{aligned}$$

(b) is equivalent aan (a) vanwege van de definitie van M, (c) is equivalent aan (b) vanwege de definitie van een NDHA.

Stap:

$$\begin{aligned}
& n \to a\langle u_1 \ldots u_n\rangle && \text{(a)} \\
& \text{desda } \exists_{n_1 \ldots n_n, r}(n_1 \ldots n_n \in L(r) \wedge n \to a\langle r\rangle \wedge n_1 \to u_1 \wedge \ldots \wedge n_n \to u_n) && \text{(b)} \\
& \text{desda } \exists_{n_1 \ldots n_n}(n \in \alpha'(a, n_1 \ldots n_n) \wedge n_1 \to u_1 \wedge \ldots \wedge n_n \to u_n) && \text{(c)} \\
& \text{desda } \exists_{n_1 \ldots n_n}(n \in \alpha'(a, n_1 \ldots n_n) \wedge n_1 \in M\|u_1 \wedge \ldots \wedge n_n \in M\|u_n) && \text{(d)} \\
& \text{desda } n \in M\|a\langle u_1 \ldots u_n\rangle && \text{(e)}
\end{aligned}$$

(b) is equivalent aan (a) vanwege de definitie van een RHG, (c) is equivalent aan (b) vanwege van de definitie van M, (d) is equivalent aan (c) vanwege de inductie hypothese, (e) is equivalent aan (d) vanwege van de 'subhedge2' stap van een NDHA.

Vervolgens laten we zien dat voor elke hedge $u_1 \dots u_n$ (waarbij elke u_i een tree is) deze in $L(G)$ zit dan en slechts dan deze ook in $L(M)$ zit:

$$
\begin{aligned}
& u_1 \dots u_n \in L(G) && \text{(a)}\\
& \text{desda } r_f \twoheadrightarrow u_1 \dots u_n && \text{(b)}\\
& \text{desda } \exists_{n_1 \dots n_n}(n_1 \dots n_n \in L(r_f) \wedge n_1 \twoheadrightarrow u_1 \wedge \dots \wedge n_n \twoheadrightarrow u_n) && \text{(c)}\\
& \text{desda } \exists_{n_1 \dots n_n}(n_1 \dots n_n \in F' \wedge n_1 \in M\|u_1 \wedge \dots \wedge n_n \in M\|u_n) && \text{(d)}\\
& \text{desda } (M\|u_1 \dots u_n) \cap F' \neq \emptyset && \text{(e)}\\
& \text{desda } u_1 \dots u_n \in L(M) && \text{(f)}
\end{aligned}
$$

(b) is equivalent aan (a) vanwege de definitie van een RHG, (c) is equivalent aan (b) vanwege de definitie van een RHG, (d) is equivalent aan (c) vanwege bovenstaand lemma en de definitie van M, (e) is equivalent aan (d) vanwege de 'concatenatie2' stap van een NDHA, (f) is equivalent aan (e) vanwege de definitie van een NDHA. Dus $L(G) = L(M)$. □

In sectie 4.2.3 is een voorbeeld te vinden hoe een RHG naar een NDHA wordt geconverteerd.

4.3.2 NDHA ⇒ RHG

Stelling 4.3.2. *Bij iedere NDHA M is een RHG G te maken zodat $L(M) = L(G)$.*

Bewijs. Gegeven een NDHA $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, maak een RHG $G = \langle \Sigma', X', N', P', r'_f \rangle$, waarbij:

$$
\begin{aligned}
\Sigma' &= \Sigma\\
X' &= X\\
N' &= Q\\
(n \to x) \in P' &\leftrightarrow n \in \iota(x)\\
(n \to a\langle r \rangle) \in P' &\leftrightarrow r \text{ is een reguliere expressie zodat:}\\
&\quad \forall_{\bar{q} \in Q^*} \bar{q} \in L(r) \leftrightarrow n \in \alpha(a, \bar{q})\\
r'_f &:= \text{een reguliere expressie zodat:}\\
&\quad L(r'_f) = F
\end{aligned}
$$

Op dezelfde manier als bij het bewijs RHG ⇒ NDHA is te laten zien dat $L(M) = L(G)$. □

Neem de volgende NDHA, $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, welke binaire bomen met a of b als knoop en x als blad accepteert als voorbeeld:

$$
\begin{aligned}
\Sigma &= \{a, b\}\\
X &= \{x\}\\
Q &= \{q_x\}\\
\iota(x) &= \{q_x\}\\
\alpha(a, \bar{q}) &\ni q_x \quad \text{als } \bar{q} \in L(q_x q_x)\\
\alpha(b, \bar{q}) &\ni q_x \quad \text{als } \bar{q} \in L(q_x q_x)\\
F &= L(q_x)
\end{aligned}
$$

Hierbij is de volgende RHG, $G = \langle \Sigma, X, N, P, r_f \rangle$, te maken, waarbij:

$$\begin{array}{rcl} \Sigma & = & \{a, b\} \\ X & = & \{x\} \\ N & = & \{q_x\} \\ P & = & \left\{ \begin{array}{lcl} q_x & \rightarrow & a\langle q_x q_x \rangle, \\ q_x & \rightarrow & b\langle q_x q_x \rangle, \\ q_x & \rightarrow & x \end{array} \right\} \\ r_f & = & q_x \end{array}$$

4.3.3 DHA ⇒ NDHA

Stelling 4.3.3. *Bij iedere DHA M kan een NDHA M' gemaakt worden zodat $L(M) = L(M')$.*

Bewijs. Gegeven een DHA $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, maak een NDHA $M' = \langle \Sigma', X', Q', \alpha', \iota', F' \rangle$, waarbij:

$$\begin{array}{rcl} \Sigma' & = & \Sigma \\ X' & = & X \\ Q' & = & Q \\ \iota'(x) & = & \{\iota(x)\} \\ \alpha'(a, \bar{q}) & = & \{\alpha(a, \bar{q})\} \\ F' & = & F \end{array}$$

Kortom, een DHA is een speciaal geval van een NDHA, namelijk waarbij het resultaat van de functies ι en α een verzameling met precies één state is. □

De DHA welke binaire bomen accepteert beschreven in sectie 4.2.1 is dan ook eenvoudig om te schrijven naar een NDHA, $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$:

$$\begin{array}{rcl} \Sigma & = & \{b\} \\ X & = & \{x\} \\ Q & = & \{q_x, q_b\} \\ \iota(x) & = & \{q_x\} \\ \alpha(b, \bar{q}) & \ni & q_b \quad \text{als } \bar{q} \in L((q_b|q_x)(q_b|q_x)) \\ F & = & L(q_b) \end{array}$$

Merk op dat de q_0 state ontbreekt, dit komt omdat een NDHA een verzameling oplevert en daardoor dus ook een lege verzameling op kan leveren. Dit in tegenstelling tot een DHA, welke altijd iets moet opleveren.

4.3.4 NDHA ⇒ DHA

Stelling 4.3.4. *Bij iedere NDHA M kan een DHA M' gemaakt worden zodat $L(M) = L(M')$.*

Bewijs. Gegeven een NDHA $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, maak een DHA $M' = \langle \Sigma', X', Q', \alpha', \iota', F' \rangle$, waarbij:

$$\begin{aligned} \Sigma' &= \Sigma \\ X' &= X \\ Q' &= 2^Q \\ \iota'(x) &= \iota(x) \\ \alpha'(a, q'_1 \dots q'_k) &= \bigcup \{\alpha(a, q_1 \dots q_k) \mid q_1 \in q'_1, \dots, q_k \in q'_k\} \\ F' &= \{q'_1 \dots q'_k \mid q_1 \dots q_k \in F, q_1 \in q'_1, \dots, q_k \in q'_k\} \end{aligned}$$

Om te bewijzen dat $L(M) = L(M')$ bewijzen we eerst het volgende lemma: $\boxed{q \in M\|u \text{ desda } q \in M'\|u}$ voor elke tree u en state $q \in Q$. Dit gaat met behulp van inductie.

Basis:

$$\begin{aligned} & q \in M\|x && \text{(a)} \\ \text{desda } & q \in \iota(x) && \text{(b)} \\ \text{desda } & q \in \iota'(x) && \text{(c)} \\ \text{desda } & q \in M'\|x && \text{(d)} \end{aligned}$$

(b) is equivalent aan (a) vanwege van de definitie van een DHA, (c) is equivalent aan (b) vanwege van de definitie van M', (d) is equivalent aan (c) vanwege van de definitie van een NDHA.

Stap:

$$\begin{aligned} & q \in M\|a\langle u_1 \dots u_n \rangle && \text{(a)} \\ \text{desda } & \exists_{q_1 \dots q_n}(q \in \alpha(a, q_1 \dots q_n) \wedge q_1 \in M\|u_1 \wedge \dots \wedge q_n \in M\|u_n) && \text{(b)} \\ \text{desda } & \exists_{q_1 \dots q_n}(q \in \alpha(a, q_1 \dots q_n) \wedge q_1 \in M'\|u_1 \wedge \dots \wedge q_n \in M'\|u_n) && \text{(c)} \\ \text{desda } & q \in \bigcup \{\alpha(a, q_1 \dots q_k) \mid q_1 \in M'\|u_1, \dots, q_n \in M'\|u_n\} && \text{(d)} \\ \text{desda } & q \in \alpha'(a, M'\|u_1 \dots M'\|u_n) && \text{(e)} \\ \text{desda } & q \in M'\|a\langle u_1 \dots u_n \rangle && \text{(f)} \end{aligned}$$

(b) is equivalent aan (a) vanwege de 'subhedge2' stap van een NDHA, (c) is equivalent aan uit (b) vanwege de inductie hypothese, (d) is equivalent aan (c) vanwege de definitie van M', (e) is equivalent aan (d) vanwege de 'subhedge2' stap van een DHA.

Vervolgens laten we zien dat voor elke hedge $u_1 \dots u_n$ (waarbij elke u_i een tree is) deze in $L(M)$ zit dan en slechts dan deze ook in $L(M')$ zit:

$$u_1 \dots u_n \in L(M) \quad \text{(a)}$$
$$\text{desda } (M\|u_1 \dots u_n) \cap F \neq \emptyset \quad \text{(b)}$$
$$\text{desda } \{q_1 \dots q_n \mid q_1 \in M\|u_1, \dots, q_n \in M\|u_n\} \cap\ F \neq \emptyset \quad \text{(c)}$$
$$\text{desda } \exists_{q_1 \dots q_n}(q_1 \dots q_n \in F \wedge q_1 \in M\|u_1 \wedge \dots \wedge q_n \in M\|u_n) \quad \text{(d)}$$
$$\text{desda } \exists_{q_1 \dots q_n}(q_1 \dots q_n \in F \wedge q_1 \in M'\|u_1 \wedge \dots \wedge q_n \in M'\|u_n) \quad \text{(e)}$$
$$\text{desda } (M'\|u_1 \dots M'\|u_n) \in \{q'_1 \dots q'_k \mid q_1 \dots q_k \in F, q_1 \in q'_1, \dots q_k \in q'_k\} \quad \text{(f)}$$
$$\text{desda } M'\|u_1 \dots u_n \in F' \quad \text{(g)}$$
$$\text{desda } u_1 \dots u_n \in L(M') \quad \text{(h)}$$

(b) is equivalent aan (a) vanwege de definitie van een NDHA, (c) is equivalent aan (b) vanwege de 'concatenatie2' stap van een NDHA, het is triviaal dat (d) equivalent is aan (c), (e) is equivalent aan (d) vanwege bovenstaand lemma, het is triviaal dat (f) equivalent is aan (e), (g) is equivalent aan (f) vanwege de 'concatenatie2' stap van een DHA, (h) is equivalent aan (g) vanwege de definitie van een DHA. Dus $L(M) = L(M')$ □

De NDHA uit sectie 4.2.3 welke enkel binaire bomen met een maximale diepte van 3 toestaat is te vertalen naar de volgende DHA, $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$:

$$\begin{aligned}
\Sigma &= \{b\} \\
X &= \{x\} \\
Q &= \{\emptyset, \{q_{b1}\}, \{q_{b1}, q_{b2}\}, \{q_{b1}, q_{b2}, q_{b3}\}, \{q_x\}\} \\
\iota(x) &= \{q_x\} \\
\alpha(b, \bar{q}) &= \begin{cases} \{q_{b1}\} & \text{als } \bar{q} \in L((\{q_{b1}, q_{b2}\}|\{q_{b1}, q_{b2}, q_{b3}\}|\{q_x\})^2) \\ \{q_{b1}, q_{b2}\} & \text{als } \bar{q} \in L((\{q_{b1}, q_{b2}, q_{b3}\}|\{q_x\})^2) \\ \{q_{b1}, q_{b2}, q_{b3}\} & \text{als } \bar{q} \in L(\{q_x\}^2) \\ \emptyset & \text{anders} \end{cases} \\
F &= L(\{q_{b1}\}|\{q_{b1}, q_{b2}\}|\{q_{b1}, q_{b2}, q_{b2}\}|\{q_x\})
\end{aligned}$$

Merk op dat in Q alleen de gebruikte states opgenomen zijn.

Corollarium 4.3.5. *De volgende klassen talen zijn equivalent:*

1. *De klasse talen gegenereerd door een RHG.*
2. *De klasse talen geaccepteerd door een DHA.*
3. *De klasse talen geaccepteerd door een NDHA.*

4.4 Uitgebreide RHG's

Om de vergelijking met RelaxNG gemakkelijker te maken zullen we een aantal uitbreidingen aan RHG's doen. Standaard zien de productie regels van een RHG er als volgt uit:

1. $n \to x$, waarbij n een non-terminal uit N is en x een variabele uit X is.

2. $n \to a\langle r \rangle$ waarbij a een symbool uit Σ en r een reguliere expressie gebruikmakend van de non-terminals, N, is.

De wortel, r_f, is een reguliere expressie gebruikmakend van de non-terminals uit N.

Definitie 4.4.1. *Een* RHG$^+$ *is een RHG met productieregels in de volgende vorm:*

$$n \to e$$

Hierbij is n een non-terminal uit N en e een reguliere expressie over het volgende:

1. *x, een variabele uit X.*
2. *$a\langle e' \rangle$, waarbij a een symbool uit Σ is en e' een reguliere expressie over het volgende is:*
   (a) *n, een non-terminal uit N.*
   (b) *x, een variabele uit X.*
   (c) *$b\langle e'' \rangle$, waarbij b een symbool uit Σ is en e'' een reguliere expressie over de elementen in deze lijst is.*
3. *m, waarbij m een non-terminal uit N ongelijk aan n is.*

Stelling 4.4.2. *Een RHG$^+$ is niet krachtiger dan een RHG.*

Bewijs. Met behulp van inductie is aan te tonen dat dit een RHG niet krachtiger maakt. Elke productieregel in nieuwe vorm is te vervangen door een aantal productieregels in de oude vorm. *Basis:*

1. $n \to x$, deze correspondeert met een regel in de oude vorm.
2. $n \to a\langle r \rangle$, deze correspondeert met een regel in de oude vorm.

Stap:

1. $n \to m$, verwijder deze regel en vervang in alle reguliere expressies van de productieregels en de wortel de de non-terminal n door m. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} r \twoheadrightarrow \ldots n \ldots \twoheadrightarrow \ldots m \ldots \quad &\text{als} \quad n \to m \\ r \twoheadrightarrow \ldots m \ldots \end{aligned}$$

2. $n \to e^*$, vervang deze regel door de regel $n \to e$, vervang in alle reguliere expressies van de productieregels en de wortel de de non-terminal n door (n^*). Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} r \twoheadrightarrow \ldots n \ldots \twoheadrightarrow \ldots (e^*) \ldots \quad &\text{als} \quad n \to e^* \\ r \twoheadrightarrow \ldots (n^*) \ldots \twoheadrightarrow \ldots (e^*) \ldots \quad &\text{als} \quad n \to e \end{aligned}$$

3. $n \to ef$, vervang deze regel door de regels $n_e \to e$ en $n_f \to f$, vervang in alle reguliere expressies van de productieregels en de wortel de non-terminal n door $(n_e n_f)$. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} r \twoheadrightarrow \ldots n \ldots &\twoheadrightarrow \ldots (ef) \ldots \quad \text{als} \quad n \to ef \\ r \twoheadrightarrow \ldots (n_e n_f) \ldots &\twoheadrightarrow \ldots (ef) \ldots \quad \text{als} \quad n_e \to e \text{ en } n_f \to f \end{aligned}$$

4. $n \to e|f$, vervang deze regel door de regels, $n_e \to e$ en $n_f \to f$, vervang in alle reguliere expressies van de productieregels en de wortel de non-terminal n door $(n_e|n_f)$. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} r \twoheadrightarrow \ldots n \ldots &\twoheadrightarrow \ldots (e|f) \ldots \quad \text{als} \quad n \to e|f \\ r \twoheadrightarrow \ldots (n_e|n_f) \ldots &\twoheadrightarrow \ldots (e|f) \ldots \quad \text{als} \quad n_e \to e \text{ en } n_f \to f \end{aligned}$$

5. $n \to a\langle \ldots (e^*) \ldots \rangle$, voeg de regel $n_e \to e$ toe en vervang deze regel door $n \to a\langle \ldots (n_e^*) \ldots \rangle$. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} n &\to a\langle \ldots (e^*) \ldots \rangle \\ n \to a\langle \ldots (n_e) \ldots \rangle &\twoheadrightarrow a\langle \ldots (e^*) \ldots \rangle \quad \text{als} \quad n_e \to e \end{aligned}$$

6. $n \to a\langle \ldots (ef) \ldots \rangle$, voeg de regels $n_e \to e$ en $n_f \to f$ toe en vervang deze regel door $n \to a\langle \ldots (n_e n_f) \ldots \rangle$. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} n &\to a\langle \ldots (ef) \ldots \rangle \\ n \to a\langle \ldots (n_e n_f) \ldots \rangle &\twoheadrightarrow a\langle \ldots (ef) \ldots \rangle \quad \text{als} \quad n_e \to e \text{ en } n_f \to f \end{aligned}$$

7. $n \to a\langle \ldots (e|f) \ldots \rangle$, voeg de regels $n_e \to e$ en $n_f \to f$ toe en vervang deze regel door $n \to a\langle \ldots (n_e|n_f) \ldots \rangle$. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{aligned} n &\to a\langle \ldots (e|f) \ldots \rangle \\ n \to a\langle \ldots (n_e|n_f) \ldots \rangle &\twoheadrightarrow a\langle \ldots (e|f) \ldots \rangle \quad \text{als} \quad n_e \to e \text{ en } n_f \to f \end{aligned}$$

□

Definitie 4.4.3. *Een* RHG^{+^+} *is een RHG*$^+$ *waarbij de wortel een reguliere expressie over het volgende is:*

1. *n, een non-terminal uit N.*
2. *x, een variabele uit X.*
3. *$a\langle e' \rangle$, waarbij a een symbool uit Σ is en e' een reguliere expressie over de elementen in deze lijst is.*

Stelling 4.4.4. *Een RHG*$^{+^+}$ *is niet krachtiger dan een RHG*$^+$*.*

Bewijs. Een equivalente RHG$^+$ is gemakkelijk te construeren. Gegeven de wortel $r_f = e$, voeg de productieregel $n_f \to e$ toe en verander de wortel in $r_f = n_f$. Te zien is dat de volgende herschrijvingen equivalent zijn:

$$\begin{array}{ccc} r_f \to e & & \\ r_f \to n_f \to e & \text{als} & n_f \to e \end{array}$$

□

Neem de volgende RHG$^{+^+}$, $G = \langle \Sigma, X, N, P, r_f \rangle$, als voorbeeld:

$$\begin{array}{rcl} \Sigma & = & \{a, b\} \\ X & = & \{x\} \\ N & = & \{n_x\} \\ P & = & \{n_x \to b\langle a\langle n_x^* \rangle \rangle | x\} \\ r_f & = & a\langle n_x^* \rangle \end{array}$$

Deze is met behulp van bovenstaande stappen naar de volgende RHG te herschrijven:

$$\begin{array}{rcl} \Sigma & = & \{a, b\} \\ X & = & \{x\} \\ N & = & \{n_f, n_{x1}, n_{x2}, n_a\} \\ P & = & \left\{ \begin{array}{lcl} n_f & \to & a\langle (n_{x1}|n_{x2})^* \rangle, \\ n_{x1} & \to & b\langle n_a \rangle, \\ n_a & \to & a\langle (n_{x1}|n_{x2})^* \rangle, \\ n_{x2} & \to & x \end{array} \right\} \\ r_f & = & n_f \end{array}$$

Corollarium 4.4.5. *De volgende eigenschappen maken RHG's niet krachtiger:*

1. *Een reguliere expressie over variabelen, hedges en non-terminals aan de rechterkant van de productieregels.*
2. *Een reguliere expressie over variabelen, hedges en non-terminals als wortel.*

4.5 Single typed en typeless RHG's

Om de vergelijking met XML-schema en DTD gemakkelijker te maken zullen we in deze sectie single typed en typeless RHG's bespreken. Single typed en typeless RHG's zijn gebaseerd op respectievelijk single typed EDTD's en DTD's besproken in [2].

Definitie 4.5.1. *Gegeven een RHG $G = \langle \Sigma, X, N, P, r_f \rangle$, de non-terminals $n, n' \in N$ zijn* disjunct *indien:*

$$\forall_{a,a',r,r'}((n \to a\langle r \rangle) \in P \wedge (n' \to a'\langle r' \rangle) \in P \to a \neq a')$$

Definitie 4.5.2. *Een RHG $G = \langle \Sigma, X, N, P, r_f \rangle$ is* single typed (ST) *indien:*

$$\forall_{e \in \{r_f\} \cup \{r \mid n \to a\langle r \rangle \in P\}} \forall_{n,n' \in e}(n \neq n' \to n \text{ en } n' \text{ zijn disjunct})$$

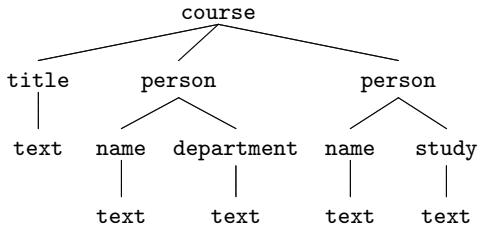
Bekijk de RHG welke roosters genereert, te vinden in sectie 4.1.1. Hier zijn 2 typen personen, namelijk:

- Een 'docent', deze heeft een naam en werkt bij nul of meerdere afdelingen.
- Een 'student', deze heeft een naam en doet één of meerdere studies.

Bij elke cursus willen we minstens éń docent en één student hebben. Dit resulteert in de volgende regels:

$$\begin{aligned} n_{course} &\rightarrow \texttt{course}\langle n_{title} {n_{teacher}}^{+} {n_{student}}^{+} \rangle \\ n_{teacher} &\rightarrow \texttt{person}\langle n_{name} {n_{department}}^{*} \rangle \\ n_{student} &\rightarrow \texttt{person}\langle n_{name} {n_{study}}^{+} \rangle \end{aligned}$$

Een voorbeeld van een hedge die deze RHG genereert is te vinden in figuur 4.1.



Figuur 4.1: Een hedge gegenereerd door de rooster RHG

Zoals te zien zijn de non-terminals $n_{student}$ en $n_{teacher}$ niet disjunct, maar komen wel samen in een reguliere expressie voor. Deze RHG is dus niet single typed.

Indien we deze RHG aanpassen met de volgende regels, is die wel single typed. $n_{student}$ en $n_{teacher}$ zijn nog steeds niet disjunct maar komen niet meer in dezelfde reguliere expressie voor.

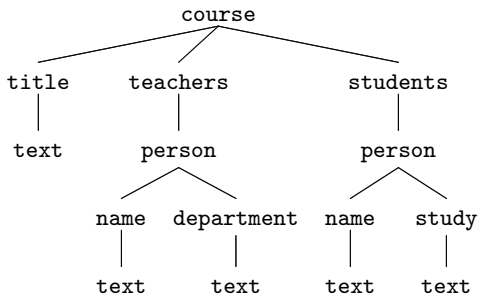
$$\begin{aligned} n_{course} &\rightarrow \texttt{course}\langle n_{title} n_{teachers} n_{student} \rangle \\ n_{teachers} &\rightarrow \texttt{teachers}\langle {n_{teacher}}^{+} \rangle \\ n_{students} &\rightarrow \texttt{students}\langle {n_{student}}^{+} \rangle \\ n_{teacher} &\rightarrow \texttt{person}\langle n_{name} {n_{department}}^{*} \rangle \\ n_{student} &\rightarrow \texttt{person}\langle n_{name} {n_{study}}^{+} \rangle \end{aligned}$$

Merk op dat dit niet enkel de RHG verandert, maar dat ook de hedges welke de RHG genereert veranderen. Zie figuur 4.2 voor een hedge die door deze ST-RHG gegenereerd is.

Definitie 4.5.3. *Gegeven een hedge u met de subhedge v, $ancestors^u(v)$ [2] zijn de symbolen die je tegenkomt als je de hedge u vanuit zijn wortel naar de subhedge v volgt.*

Neem de hedge u te zien in figuur 4.1:

$$ancestors^u(\texttt{person}\langle\texttt{name}\langle\texttt{text}\rangle\texttt{study}\langle\texttt{text}\rangle\rangle) = \{\texttt{course}\}$$



Figuur 4.2: Een hedge gegenereerd door de single typed rooster RHG

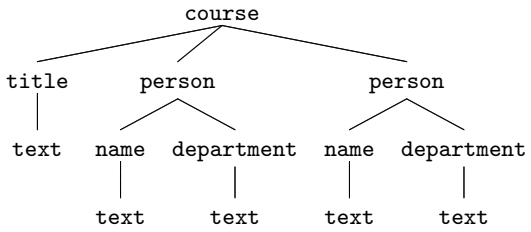
Lemma 4.5.4. *Een taal L van hedges kan gegenereerd worden door een ST-RHG [2], dan en slechts dan:*

1. *Er een RHG G is zodat $L(G) = L$.*
2. *Indien voor alle hedges $u_1, u_2 \in L$ waarbij $a\langle v \rangle$ een subhedge van u_1 is en $a\langle w \rangle$ een subhedge van u_2 is zodat $ancestors^{u_1}(a\langle v \rangle) = ancestors^{u_2}(a\langle w \rangle)$ dan zit de hedge u_1 waarbij de subhedge $a\langle v \rangle$ vervangen is door $a\langle w \rangle$ ook in L.*

Het is eenvoudig in te zien dat klopt. Alle non-terminals die in een reguliere expressie worden gebruikt zijn disjunct, dit betekent dat tijdens het pad van de wortel naar elke subhedge je voor elk symbool telkens maar de keuze uit één non-terminal hebt. Subhedges welke uit de zelfde non-terminal zijn gegenereerd mogen daarom altijd omgeruild worden.

Stelling 4.5.5. *Niet voor elke RHG G is een ST-RHG G' te maken zodat $L(G) = L(G')$.*

Het is eenvoudig in te zien dat voor de RHG die roosters genereert uit sectie 4.1.1 geen ST-RHG te maken is. De hedge te zien in figuur 4.3, waarbij de ene subhedge `person` vervangen is door de andere subhedge `person`, zit niet meer in de taal van deze RHG.



Figuur 4.3: Een hedge welke je verkrijgt door subhedge vervanging bij een hedge gegenereerd door de rooster RHG

Definitie 4.5.6. *Een RHG $G = \langle \Sigma, X, N, P, r_f \rangle$ is* typeless (TL) *indien:*

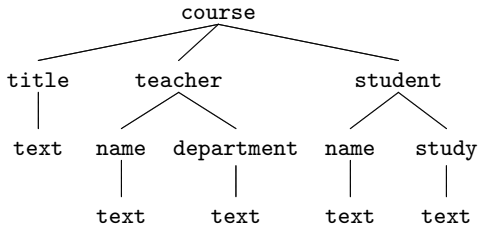
$$\forall_{n,n' \in N}(n \neq n' \rightarrow n \text{ en } n' \text{ zijn disjunct})$$

Stelling 4.5.7. *Als een RHG typeless is dan is die single typed.*

Bovenstaande RHG's zijn niet typeless. Gezien de non-terminals $n_{student}$ en $n_{teacher}$ disjunct zijn. Nu passen de RHG aan met de volgende productieregels:

$$\begin{aligned} n_{course} &\rightarrow \texttt{course}\langle n_{title} {n_{teacher}}^{+} {n_{student}}^{+} \rangle \\ n_{teacher} &\rightarrow \texttt{teacher}\langle n_{name} {n_{department}}^{*} \rangle \\ n_{student} &\rightarrow \texttt{student}\langle n_{name} {n_{study}}^{+} \rangle \end{aligned}$$

Merk op dat dit niet enkel de RHG verandert, maar dat ook de hedges die de RHG genereert veranderen. Zie hiervoor figuur 4.4 voor een hedge die door deze ST-RHG gegenereerd is.



Figuur 4.4: Een hedge gegenereerd door de typeless rooster RHG

Lemma 4.5.8. *Een taal L van hedges kan gegenereerd worden door een TL-RHG [2], dan en slechts dan:*

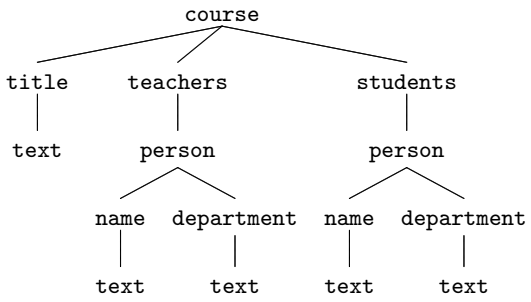
1. *Er een RHG G is zodat $L(G) = L$.*
2. *Indien voor alle hedges $u_1, u_2 \in L$ waarbij $a\langle v \rangle$ een subhedge van u_1 is en $a\langle w \rangle$ een subhedge van u_2 is, dan zit de hedge u_1 waarbij de subhedge $a\langle v \rangle$ vervangen is door $a\langle w \rangle$ ook in L.*

Het is eenvoudig in te zien dat klopt. Alle non-terminals die in een TL-RHG voorkomen zijn disjunct, dit betekent dat alle subhedges met hetzelfde symbool uit dezelfde non-terminal gegenereerd zijn. Subhedges welke uit de zelfde non-terminal zijn gegenereerd mogen altijd omgeruild worden.

Stelling 4.5.9. *Niet voor elke ST-RHG G is een TL-RHG G' te maken zodat $L(G) = L(G')$.*

Het is eenvoudig in te zien dat voor de voorgaande ST-RHG geen TL-RHG te maken is. De hedge te zien in figuur 4.5, waarbij de ene subhedge `person` vervangen is door de andere subhedge `person`, zit niet meer in de taal van die RHG.

Zie de RHG die binaire bomen met een maximale diepte van 3 accepteert, gegeven in sectie 4.2.3, voor een ander voorbeeld van een RHG die wel single typed maar niet typeless is. Ook voor dit voorbeeld is geen RHG te geven die typeless is.



Figuur 4.5: Een hedge welke je verkrijgt door subhedge vervanging bij een hedge gegenereerd door de single typed rooster RHG

4.6 Afsluitingseigenschappen

In deze sectie zullen we de afsluitingseigenschappen van de klassen talen genereert door RHG's, ST-RHG en TL-RHG's bespreken. Dit is ook te lezen in [5], echter ontbreken constructies om de verschillende afsluitingseigenschappen te verwezenlijken en bewijzen in dat artikel.

Stelling 4.6.1. *Gegeven twee DHA's M_1 en M_2 is er een DHA M te maken met als taal $L(M) = L(M_1) \cup L(M_2)$.*

Bewijs. Gegeven een DHA $M_1 = \langle \Sigma_1, X_1, Q_1, \alpha_1, \iota_1, F_1 \rangle$ en een DHA $M_2 = \langle \Sigma_2, X_2, Q_2, \alpha_2, \iota_1, F_2 \rangle$ zodat $Q_1 \cap Q_1 = \emptyset$ maak de DHA $M = \langle \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, Q_1 \cup Q_2, \alpha_1 \cup \alpha_2, \iota_1 \cup \iota_2, F_1 \cup F_2 \rangle$. We laten nu zien dat voor elke hedge u deze in $L(M)$ zit dan en slechts dan deze in $L(M_1) \cup L(M_2)$ zit.

$$
\begin{aligned}
& u \in L(M) \\
\text{desda } & M \| u \in F_1 \cup F_2 \\
\text{desda } & M \| u \in F_1 \vee M \| u \in F_2 \\
\text{desda } & u \in L(M_1) \vee u \in L(M_2) \\
\text{desda } & u \in L(M_1) \cup L(M_2)
\end{aligned}
$$

Dus $L(M) = L(M_1) \cup L(M_2)$. □

Stelling 4.6.2. *Gegeven een DHA M_1 is er een DHA M te maken met als taal $L(M) = \overline{L(M_1)}$.*

Bewijs. Gegeven een DHA $M_1 = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$ maak de DHA $M = \langle \Sigma, X, Q, \alpha, \iota, \overline{F} \rangle$. We laten nu zien dat voor elke hedge u deze in $L(M)$ zit dan en slechts dan deze niet in $L(M_1)$ zit.

$$
\begin{aligned}
& u \in L(M) \\
\text{desda } & M \| u \in F \\
\text{desda } & M \| u \notin \overline{F} \\
\text{desda } & u \notin L(M_1)
\end{aligned}
$$

□

Dus $L(M) = \overline{L(M_1)}$.

Stelling 4.6.3. *Gegeven twee DHA's M_1 en M_2 is er een DHA M te maken met als taal $L(M) = L(M_1) \cap L(M_2)$.*

Bewijs. Gegeven een NDHA $M_1 = \langle \Sigma, X, Q_1, \alpha_1, \iota_1, F_1 \rangle$ en een NDHA $M_2 = \langle \Sigma, X, Q_2, \alpha_2, \iota_1, F_2 \rangle$ maak de NDHA $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$, waarbij:

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \alpha(a, (p_1, q_1) \dots (p_n, q_n)) &= \{(p, q) \mid p \in \alpha_1(a, p_1 \dots p_n), q \in \alpha_2(a, q_1 \dots q_n)\} \\ \iota(x) &= \{(p, q) \mid p \in \iota_1(x), q \in \iota_2(x)\} \\ F &= \{(p_1, q_1) \dots (p_n, q_n) \mid p_1 \dots p_n \in F_1, q_1 \dots q_n \in F_2\} \end{aligned}$$

Zoals te zien bestaan de states van deze NDHA uit tupels van states uit de twee NDHA's. Indien we deze NDHA een hedge laten accepteren zal die feitelijk de twee oorspronkelijke NDHA's parallel laten lopen.

Om te bewijzen dat $L(M) = L(M_1) \cap L(M_2)$ bewijzen we eerst het volgende lemma: $\boxed{(p, q) \in M \| u \leftrightarrow p \in M_1 \| u \wedge q \in M_2 \| u}$ voor elke tree u en state p en q. Dit gaat met behulp van inductie.

Basis:

$$\begin{aligned} & (p, q) \in M \| x && \text{(a)} \\ \text{desda } & (p, q) \in \iota(x) && \text{(b)} \\ \text{desda } & p \in \iota_1(x) \wedge p \in \iota_2(x) && \text{(c)} \\ \text{desda } & p \in M_1 \| x \wedge p \in M_2 \| x && \text{(d)} \end{aligned}$$

(b) is equivalent aan (a) vanwege van de definitie van een NDHA, (c) is equivalent aan (b) vanwege van de definitie van M, (d) is equivalent aan (c) vanwege van de definitie van een NDHA.

Stap:

$$\begin{aligned} & (p, q) \in M \| a\langle u_1 \dots u_n \rangle && \text{(a)} \\ \text{desda } & \exists_{(p_1,q_1)\dots(p_n,q_n)}((p, q) \in \alpha(a, (p_1, q_1) \dots (p_n, q_n)) \wedge \\ & \quad (p_1, q_1) \in M \| u_1 \wedge \dots \wedge (p_n, q_n) \in M \| u_n) && \text{(b)} \\ \text{desda } & \exists_{(p_1,q_1)\dots(p_n,q_n)}((p, q) \in \alpha(a, (p_1, q_1) \dots (p_n, q_n)) \wedge \\ & \quad p_1 \in M_1 \| u_1 \wedge \dots \wedge p_n \in M_1 \| u_n \wedge q_1 \in M_2 \| u_1 \wedge \dots \wedge q_n \in M_2 \| u_n) && \text{(c)} \\ \text{desda } & \exists_{(p_1,q_1)\dots(p_n,q_n)}(p \in \alpha_1(a, p_1 \dots p_n) \wedge q \in \alpha_1(a, q_1 \dots q_n) \wedge \\ & \quad p_1 \in M_1 \| u_1 \wedge \dots \wedge p_n \in M_1 \| u_n \wedge q_1 \in M_2 \| u_1 \wedge \dots \wedge q_n \in M_2 \| u_n) && \text{(d)} \\ \text{desda } & \exists_{p_1 \dots p_n}(p \in \alpha_1(a, p_1 \dots p_n) \wedge p_1 \in M_1 \| u_1 \wedge \dots \wedge p_n \in M_1 \| u_n) \wedge \\ & \quad \exists_{q_1 \dots q_n}(q \in \alpha_2(a, q_1 \dots q_n) \wedge q_1 \in M_2 \| u_1 \wedge \dots \wedge q_n \in M_2 \| u_n) && \text{(e)} \\ \text{desda } & p \in M_1 \| a\langle u_1 \dots u_n \rangle \wedge q \in M_2 \| a\langle u_1 \dots u_n \rangle && \text{(f)} \end{aligned}$$

(b) is equivalent aan (a) vanwege de 'subhedge2' stap van een NDHA, (c) is equivalent aan uit (b) vanwege de inductie hypothese, (d) is equivalent aan (c) vanwege de definitie van M, het is triviaal dat (e) equivalent is aan (d), (f) is equivalent aan (e) vanwege de 'subhedge2' stap van een NDHA.

Vervolgens laten we zien dat voor elke hedge $u_1 \dots u_n$ (waarbij elke u_i een tree is) deze in $L(M)$ zit dan en slechts dan deze ook in $L(M_1)$ en $L(M_2)$ zit:

$$
\begin{aligned}
& u_1 \dots u_n \in L(M) && \text{(a)}\\
\text{desda } & (M\|u_1 \dots u_n) \cap F \neq \emptyset && \text{(b)}\\
\text{desda } & \{(p_1q_1)\dots(p_nq_n) \mid (p_1q_1) \in M\|u_1, \dots, (p_nq_n) \in M\|u_n\} \cap\ F \neq \emptyset && \text{(c)}\\
\text{desda } & \exists_{(p_1q_1)\dots(p_nq_n)}((p_1q_1)\dots(p_nq_n) \in F \wedge \\
& \quad (p_1q_1) \in M\|u_1 \wedge \dots \wedge (p_nq_n) \in M\|u_n) && \text{(d)}\\
\text{desda } & \exists_{(p_1q_1)\dots(p_nq_n)}(p_1 \dots p_n \in F_1 \wedge q_1 \dots q_n \in F_2 \wedge \\
& \quad (p_1q_1) \in M\|u_1 \wedge \dots \wedge (p_nq_n) \in M\|u_n) && \text{(e)}\\
\text{desda } & \exists_{(p_1q_1)\dots(p_nq_n)}(p_1 \dots p_n \in F_1 \wedge q_1 \dots q_n \in F_2 \wedge \\
& \quad p_1 \in M_1\|u_1 \wedge \dots \wedge p_n \in M_1\|u_n \wedge q_1 \in M_2\|u_1 \wedge \dots \wedge q_n \in M_2\|u_n) && \text{(f)}\\
\text{desda } & \exists_{p_1\dots p_n}(p_1 \dots p_n \in F_1 \wedge p_1 \in M_1\|u_1 \wedge \dots \wedge p_n \in M_1\|u_n) \wedge \\
& \quad \exists_{q_1\dots q_n}(q_1 \dots q_n \in F_2 \wedge q_1 \in M_2\|u_1 \wedge \dots \wedge q_n \in M_2\|u_n) && \text{(g)}\\
\text{desda } & \{p_1 \dots p_n \mid p_1 \in M_1\|u_1 \wedge \dots \wedge p_n \in M_1\|u_n\} \neq \emptyset \wedge \\
& \quad \{p_1 \dots p_n \mid p_1 \in M_2\|u_1 \wedge \dots \wedge p_n \in M_2\|u_n\} \neq \emptyset && \text{(h)}\\
\text{desda } & (M_1\|u_1 \dots u_n) \neq \emptyset \wedge (M_2\|u_1 \dots u_n) \neq \emptyset && \text{(i)}\\
\text{desda } & u_1 \dots u_n \in L(M_1) \wedge u_1 \dots u_n \in L(M_2) && \text{(j)}\\
\text{desda } & u_1 \dots u_n \in L(M_1) \cap L(M_2) && \text{(k)}
\end{aligned}
$$

□

(b) is equivalent aan (a) vanwege de definitie van een NDHA, (c) is equivalent aan (b) vanwege de 'concatenatie2' stap van een NDHA, het is triviaal dat (d) equivalent is aan (c), (e) is equivalent aan (d) vanwege de definitie van M (f) is equivalent aan (e) vanwege bovenstaand lemma, het is triviaal dat (g) equivalent is aan (f) en dat (h) equivalent is aan (g), (i) is equivalent aan (h) vanwege de 'concatenatie2' stap van een NDHA, (j) is equivalent aan (i) vanwege de definitie van een DHA, het is triviaal dat (k) equivalent is aan (j). Dus $L(M) = L(M_1) \cap L(M_2)$.

Corollarium 4.6.4. *De klasse van talen die door een RHG gegeneerd of door een (N)DHA geaccepteerd worden zijn gesloten onder de vereniging, de intersectie en het complement.*

Stelling 4.6.5. *Gegeven twee TL of ST RHG's G_1 en G_2 is er niet altijd een TL of ST RHG G te maken zodat $L(G) = L(G_1) \cup L(G_2)$.*

Bewijs. Gegeven de typeless RHG $G_1 = \langle \Sigma_1, X_1, N_1, P_1, r_{f1} \rangle$ en de typeless RHG $G_2 = \langle \Sigma_2, X_2, N_2, P_2, r_{f2} \rangle$:

$$
\begin{aligned}
\Sigma_1 &= \{a\}\\
X_2 &= \{x\}\\
N_1 &= \{n_x, n_1\}\\
P_1 &= \{n_x \to x, n_1 \to a\langle n_x \rangle\}\\
r_{f1} &= n_x n_1
\end{aligned}
\qquad
\begin{aligned}
\Sigma_2 &= \{a\}\\
X_2 &= \{y\}\\
N_2 &= \{n_y, n_2\}\\
P_2 &= \{n_y \to y, n_2 \to a\langle n_y \rangle\}\\
r_{f2} &= n_y n_2
\end{aligned}
$$

Indien we hier de vereniging van willen nemen betekend dit dat de hegdes $xa\langle x \rangle$ en $ya\langle y \rangle$ in de taal $L(G_1) \cup L(G_2)$ zitten. Mochten we hier een ST of

TL-RHG voor willen maken zou dat betekenen dat $xa\langle y\rangle$ ook in de taal moet zitten, maar deze zit er niet in. Dit duidt erop dat de resulterende RHG niet ST nog TL is.

□

Stelling 4.6.6. *Gegeven een TL of ST RHG's G_1 is er niet altijd een TL of ST RHG G te maken zodat $L(G) = \overline{L(G_1)}$.*

Bewijs. Gegeven de volgende RHG $G = \langle \Sigma, X, N, P, r_f \rangle$, waarbij:

$$\begin{aligned} \Sigma &= \{a\} \\ X &= \{x\} \\ N &= \{n_x, n_a\} \\ P &= \{n_x \to x, n_a \to a\langle n_x \rangle\} \\ r_f &= n_a \end{aligned}$$

Indien we hier de complement van willen nemen betekend dit dat de hegdes $a\langle xx\rangle$ en $a\langle x\rangle a\langle x\rangle$ in de taal $\overline{L(G_1)}$ zitten. Mochten we hier een ST-RHG voor willen maken zou dit betekenen dat $a\langle x\rangle$ ook in de taal moet zitten, maar deze zit er niet in. Dit duidt erop dat de resulterende RHG niet ST nog TL is. □

Stelling 4.6.7. *Gegeven twee TL of ST RHG's G_1 en G_2 is er respectievelijk een TL of ST RHG G te maken met als taal $L(G) = L(G_1) \cap L(G_2)$.*

Indien je de intersectie van twee TL of ST-RHG's wilt nemen moet je het volgende doen:

- De RHG's transformeren naar een NDHA.
- De intersectie van deze NDHA's nemen.
- Deze NDHA transformeren naar een RHG.

Het bewijs dat dit een TL respectievelijk ST-RHG oplevert is helaas achterwegen gelaten.

Corollarium 4.6.8. *De klasse talen welke door een single typed of typeless RHG gegeneerd worden zijn gesloten onder de intersectie, maar niet onder de vereniging en het complement.*

4.7 Strings

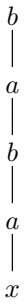
In deze sectie wordt de mogelijkheid besproken om strings te accepteren en genereren met behulp van RHG's en (N)DHA's. Dit wordt verder niet gebruikt voor de vergelijking tussen de XML schema's.

De meest simpele manier om een hedge te gebruiken om strings te representeren is het maken van een hedge met diepte nul bestaande uit enkel variabelen.

Gezien r_f, de wortel van een RHG, een reguliere expressie is, zijn op deze manier alle reguliere talen te genereren. Hierdoor zijn met een (N)DHA ook alle reguliere (string) talen te accepteren.

$a\ b\ a\ b$

Figuur 4.6: De string *abab* als een hedge bestaande uit enkel variabelen



Figuur 4.7: De string *abab* als een hedge van breedte 1 met x als beginmarkering

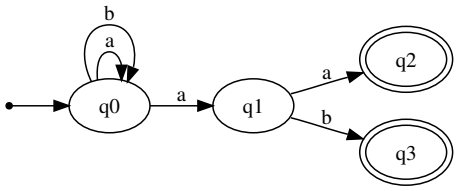
Een andere manier is door gebruik te maken van een diepe hedge met breedte één. Waarbij de string op zijn kop te lezen is, met een variabele x als begin markering.

Ook op deze manier zijn alle reguliere talen te accepteren. Elke eindige non-deterministische automaat $M_E = \langle Q, \Sigma, \delta, q_o, F\rangle$[1] is te converteren naar een NDHA $M_{ND} = \langle \Sigma', X', Q', \alpha', \iota', F'\rangle$. Waarbij:

$$\begin{aligned} \Sigma' &= \Sigma \\ X' &= \{x\} \\ Q' &= Q \\ \alpha'(a,q) &= \delta(q,a) \\ \iota'(x) &= q_0 \\ F' &= F \end{aligned}$$

Gezien een eindige automaat alle reguliere talen kan accepteren, kan een (N)DHA dat middels bovenstaande constructie ook en kan een RHG dus alle reguliere talen genereren.

Neem de volgende Glushkov automaat welke strings volgens de reguliere expressie $(a|b)^*a(a|b)$ accepteert als voorbeeld:



[1] Zie appendix A voor de definitie van een eindige automaat.

De bijbehorende NDHA is $M = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$:

$$\begin{aligned}
\Sigma &= \{a, b\} \\
X &= \{x\} \\
Q &= \{q_0, q_1, q_2, q_3\} \\
\iota(x) &= \{q_0\} \\
\alpha(a, q_0) &= \{q_0, q_1\} \\
\alpha(b, q_0) &= \{q_0\} \\
\alpha(a, q_1) &= \{q_2\} \\
\alpha(b, q_1) &= \{q_3\} \\
\alpha(l, q_2) &= \emptyset \\
\alpha(l, q_3) &= \emptyset \\
F &= \{q_2, q_3\}
\end{aligned}$$

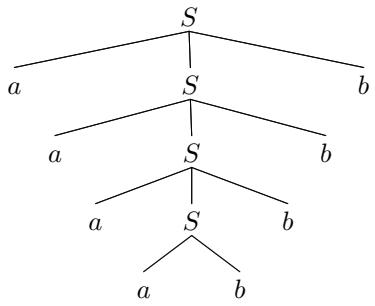
De string *abab* wordt als volgt geaccepteerd:

$$\dfrac{\dfrac{\dfrac{\dfrac{M\|x = \{q_0\}}{M\|a\langle x\rangle = \{q_0, q_1\}}}{M\|b\langle a\langle x\rangle\rangle = \{q_0, q_3\}}}{M\|a\langle b\langle a\langle x\rangle\rangle\rangle = \{q_0, q_1\}}}{M\|b\langle a\langle b\langle a\langle x\rangle\rangle\rangle\rangle = \{q_0, q_3\}}$$

Merk op dat de afleiding gelijk is aan het pad door de eindige automaat.

Lemma 4.7.1. *Een (N)DHA kan elke reguliere taal over strings accepteren. Met behulp van een RHG is elke reguliere taal over strings te genereren.*

Een andere manier is om een hedge te gebruiken als parseerboom van een taal. In dit geval representeren we de string als bladeren van de hedge, gelezen van links naar rechts. De symbolen worden genegeerd.



Figuur 4.8: De string *aaaabbbb* gerepresenteerd als bladeren van een hedge

Op deze manier zijn context vrije talen te genereren. Elke context vrije grammatica $G_{CFG} = \langle V, \Sigma, R, S \rangle$[2] is om te schrijven naar een RHG $G_{RHG} = \langle \Sigma', X', N', P', r'_f \rangle$. In dit geval komen de non-terminals van de CFG overeen

[2]Zie appendix A voor de definitie van een context vrije grammatica.

met de symbolen in de RHG en komen de symbolen uit de CFG overeen met de variabelen in de RHG. De transformatie gaat als volgt:

$$\begin{aligned}
\Sigma' &= V \\
X' &= \Sigma \\
N' &= \{n_x \mid x \in \Sigma \cup V\} \\
P' &= \{n_s \to s\langle r\rangle \mid s \to r \in R\} \cup \{n_a \to a \mid a \in X\} \\
r'_f &= n_S
\end{aligned}$$

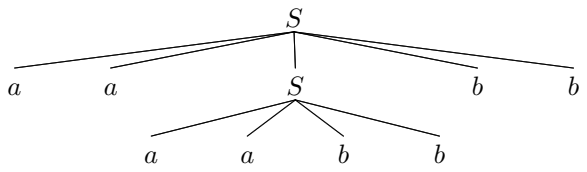
Neem de CFG $G = \langle\{S\}, \{a, b\}, R, S\rangle$ die de taal $L = \{a^n b^n \mid n \geq 1\}$ genereert:

$$R = \{S \to ab \mid aSb\}$$

Hiervoor is de volgende RHG $G = \langle\Sigma, X, N, P, r_f\rangle$ op te schrijven:

$$\begin{aligned}
\Sigma &= \{S\} \\
X &= \{a, b\} \\
N &= \{n_S, n_a, n_b\} \\
P &= \left\{\begin{array}{lcl} n_a & \to & a, \\ n_b & \to & b, \\ n_S & \to & S\langle(n_a n_b)|(n_a n_S n_b)\rangle \end{array}\right\} \\
r_f &= n_S
\end{aligned}$$

Merk op dat je op deze manier parseer bomen van een contextvrije taal genereert, een (N)DHA kan daarom dus slechts een parseerboom van een context vrije taal accepteren. Bovenstaande string kan namelijk ook op de manier in figuur 4.9 gerepresenteerd worden. De string heeft zo een andere parseerboom en zit niet in de taal van bovenstaande RHG.



Figuur 4.9: De string *aaaabbbb* op een andere manier gerepresenteerd

Merk daarnaast op dat deze RHG typeless is. Voor elke non-terminal s uit de context vrije grammatica wordt één regel $n_s \to s\langle r\rangle$ toegevoegd. Dit duidt erop dat alle non-terminals uit de RHG disjunct zijn.

Lemma 4.7.2. *Alle parseerbomen welke met een contextvrije grammatica beschreven kunnen worden, kunnen gegenereerd worden door een TL-RHG.*

Merk op dat dit voor parseerbomen niet de andere kant op werkt, de productieregel $n_S \to S\langle a^*\rangle$ is niet om te schrijven naar een regel in een CFG. Dit komt omdat een RHG meer vrijheid geeft wat betreft de rechterkant van de regels, hier mogen namelijk reguliere expressies staan.

Echter kunnen we een CFG ook strings laten genereren welke corresponderen met (de syntax van) hedges. Dit doen we door de tekens $\langle$ en $\rangle$ aan het alfabet

van de CFG toe te voegen. Eenvoudig is te zien dat op deze manier elke RHG naar een CFG herschreven kan worden. Elke productie regel $n \rightarrow a\langle r \rangle$ kan worden herschreven naar de regel $n \rightarrow a\langle n_1 \rangle$ waarbij de reguliere expressie wordt omgeschreven naar de regels n_1 tot n_n. Dit is mogelijk, want voor elke reguliere expressie is het tenslotte mogelijk een context vrije grammatica te maken. Hierbij kunnen we wel de volgende opmerkingen maken:

1. Op deze manier is het mogelijk een CFG te maken welke hedges genereert die niet correct zijn, bijvoorbeeld $a\langle\langle b \rangle\rangle$. Dit impliceert dus direct dat dit geen mooie manier is om hedges te genereren.

2. Deze manier heeft beduidend meer uitdrukkingskracht dan een RHG. Dit komt omdat er contextvrije talen kunnen worden gebruikt om de kinderen van een knoop te beschrijven. Neem de CFG $G = \langle V, \Sigma, R, S \rangle$ als voorbeeld, waarbij:

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{\langle, \rangle, a, b\} \\ R &= \left\{ \begin{array}{lcc} S & \rightarrow & ASB \mid \epsilon, \\ A & \rightarrow & a\langle S \rangle, \\ B & \rightarrow & b\langle S \rangle \end{array} \right\} \end{aligned}$$

   Dit zal hedges genereren waarbij elke knoop evenveel a kinderen als b kinderen heeft. Een RHG zal een productieregel welke er als volgt uitziet moeten bevatten:

$$n_a \rightarrow a\langle r \rangle \text{ waarbij } L(r) = \{n_a^x n_b^x \mid x \in \mathbb{N}\}$$

   Maar de reguliere expressie r is niet te maken, gezien zijn taal niet regulier is.

Hoofdstuk 5

Deterministische reguliere expressies

Gezien de reguliere expressies welke DTD en XML schema gebruiken deterministisch moeten zijn zullen we in dit hoofdstuk verder op dat begrip in gaan.

5.1 Glushkov automaten

Om een exacte defintie van deterministische reguliere expressies te kunnen maken, zullen we eerst Glushkov automaten bespreken.

Definitie 5.1.1. *Gegeven een reguliere expressie E, is op de volgende manier de* Glushkov automaat *[1], M_E welke $L(E)$ accepteert te maken. Voor een reguliere expressie E heeft M_E altijd de volgende vorm $M_E = \langle Q_E \cup \{q_0\}, \Sigma_E, \delta_E, q_0, F_E \rangle$.*

1. $M_\epsilon = \langle \{q_0\}, \emptyset, \emptyset, q_0, \{q_0\} \rangle$
2. $M_a = \langle \{q, q_0\}, \{a\}, \{q_0 \times a \to q\}, q_0, \{q\} \rangle$
3. $M_{F|G} = \langle Q_E \cup \{q_0\}, \Sigma_E, \delta_E, q_0, F_E \rangle$ *Waarbij:*

$$
\begin{aligned}
Q_E &= Q_F \cup Q_G \text{ (eventueel states hernoemd, zodat } Q_E \cap Q_F \neq \emptyset) \\
\Sigma_E &= \Sigma_F \cup \Sigma_G \\
F_E &= F_F \cup F_G \\
\delta_E(q,a) &= \begin{cases} \delta_F(q,a) & \text{indien } q \in Q_F \\ \delta_G(q,a) & \text{indien } q \in Q_G \\ \delta_F(q,a) \cup \delta_G(q,a) & \text{indien } q = q_0 \end{cases}
\end{aligned}
$$

4. $M_{FG} = \langle Q_E \cup \{q_0\}, \Sigma_E, \delta_E, q_0, F_E \rangle$ *Waarbij:*

$$Q_E = Q_F \cup Q_G \text{ (eventueel states hernoemd, zodat } Q_E \cap Q_F \neq \emptyset)$$
$$\Sigma_E = \Sigma_F \cup \Sigma_G$$
$$F_E = \begin{cases} F_F \cup (F_G \setminus \{q_0\}) & \text{indien } q_0 \in F_G \\ F_G & \text{anders} \end{cases}$$
$$\delta_E(q,a) = \begin{cases} \delta_F(q,a) & \text{indien } q \in Q_F \backslash F_F \text{ of } q = q_0 \\ \delta_G(q,a) & \text{indien } q \in Q_G \\ \delta_F(q,a) \cup \delta_G(q,a) & \text{indien } q \in F_F \end{cases}$$

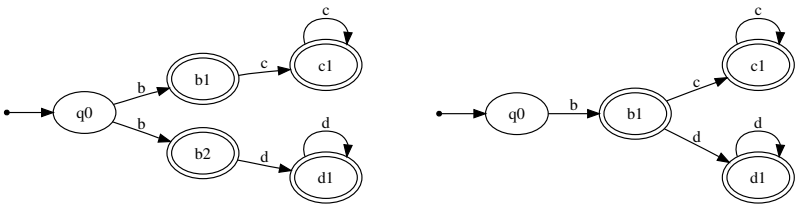
5. $M_{F^*} = \langle Q_E \cup \{q_0\}, \Sigma_E, \delta_E, q_0, F_E \rangle$ *Waarbij:*

$$Q_E = Q_F$$
$$\Sigma_E = \Sigma_F$$
$$F_E = F_F \cup \{q_1\}$$
$$\delta_E(q,a) = \begin{cases} \delta_F(q,a) \cup \delta_F(q_0,a) & \text{indien } q \in F_F \\ \delta_F(q,a) & \text{anders} \end{cases}$$

Definitie 5.1.2. *Een reguliere expressie E is* deterministisch *indien de Glushkov automaat M_E deterministisch is, of indien $L(E) = \emptyset$.*

Definitie 5.1.3. *Een reguliere taal L is* deterministisch *indien er een deterministische reguliere expressie E bestaat zodat zodat $L(E) = L$.*

Zie figuur 5.1 voor de Glushkov automaten voor de reguliere expressies $(bc^*)|(bd^*)$ en $b(c^*|d^*)$. Te zien is dat deze automaten welke dezelfde taal accepteren respectievelijk non-deterministisch en deterministisch zijn.



(a) $(bc^*)|(bd^*)$ (b) $b(c^*|d^*)$

Figuur 5.1: Glushkov automaten

Stelling 5.1.4. *Niet elke reguliere taal is deterministisch.*

Voor het bewijs van deze stelling verwijzen we naar [1]. Voor de volgende reguliere expressie $(a|b)^*a(a|b)$ is bijvoorbeeld geen deterministische equivalente reguliere expressie te maken.

Hoofdstuk 6

Vergelijking

6.1 DTD

DTD's komen overeen met een typeless RHG's met deterministische reguliere expressies en een enkel startsymbool (DTD's genereren dus enkel trees). Het converteren van de DTD

```
<!DOCTYPE root-element [element-declarations]>
```

naar de RHG, $G = \langle \Sigma, X, N, P, r_f \rangle$, gaat als volgt:

1. Voeg `#PCDATA` als variabele aan X toe en voeg `#PCDATA` als non-terminal aan N toe.
2. Voeg de regel #`PCDATA` $\rightarrow$ #`PCDATA` aan P toe.
3. Voeg voor elke regel `<!ELEMENT name (content-model)>`, `name` aan Σ en aan N toe en voeg de de productie regel `name` $\rightarrow$ `name`$\langle$`content-model`$\rangle$ aan P toe.
4. Voeg voor elke regel `<!ELEMENT name EMPTY>`, `name` aan Σ en aan N toe en voeg de de productie regel `name` $\rightarrow$ `name`$\langle\epsilon\rangle$ aan P toe.
5. $r_f =$ `root-element`

Merk hierbij op dat de variabelen overeenkomen met de basistypes, waaronder het besproken `#PCDATA`. Gezien `name` uniek is voor elk `element` is het gemakkelijk in te zien dat deze RHG typeless is.

6.2 XML schema

XML schema komt overeen met een single typed RHG met deterministische reguliere expressies en een keuze uit een enkel start element (XML schema's genereren dus enkel trees). Het grote verschil qua notatie tussen een ST-RHG en een XML schema zit hem in de productieregels. De regels van XML schema kunnen net iets anders geformaliseerd worden:

1. $n \to x$, waarbij n een non-terminal uit N is en x een variabele uit X. Merk hierbij op dat de variabelen overeenkomen met de basistypes, waaronder het besproken `xs:string`.

2. $n \to e$, waarbij n een non-terminal uit N is en e een reguliere expressie over het volgende:

   (a) $a\langle n' \rangle$, waarbij a een een symbool uit Σ is en n' een non-terminal uit N.

   Deze regel komt overeen met de volgende definitie van `element` uit XML schema, `name` correspondeert met a en `type` correspondeert met n'.

   ```
   <xs:element name="name" type="type" minOccurs="min" maxOccurs="max" />
   ```

   (b) $a\langle e' \rangle$, waarbij a een een symbool uit Σ is en e' een reguliere expressie over de elementen uit deze lijst.

   Deze regel komt overeen met de volgende definitie van `element` uit XML schema, `name` correspondeert met a en `complexType` correspondeert met e'.

   ```
   <xs:element name="name" minOccurs="min" maxOccurs="max">
     complexType
   </xs:element>
   ```

   Deze soort regel komt overeen met de definitie van `complexType` uit XML schema, `type` correspondeert met n.

   ```
   <xs:complexType name="type">
     (choice | sequence)
   </xs:complexType>
   ```

   `choice` en `sequence` komen respectievelijk overeen met de volgende reguliere expressies $(e_1|e_2|\ldots|e_n)$ en $(e_1 e_2 \ldots e_n)$.

Om te bewijzen dat dit qua uitdrukkingskracht niet sterker is dan een standaard RHG verwijzen we naar de bewijzen in sectie 4.4, op zo'n zelfde manier is te bewijzen dat al deze regels zijn te herschrijven naar regels van een standaard RHG.

De Element Declarations Consistent (EDC) zorgt er voor dat deze RHG single typed is. Deze zegt dat binnen een `complexType` geen elementen met dezelfde naam maar met verschillende types mogen voorkomen. Dit is precies dezelfde beperking als single typeness ons oplegt.

6.3 Relax NG

Relax NG komt overeen met een RHG^{+^+}. Dit is vrij triviaal, Relax NG is in feite slechts een andere syntax om een RHG^{+^+} op te schrijven. Merk hierbij op dat de variabelen overeenkomen met de basistypes, waaronder het besproken `text`.

6.4 Conclusie

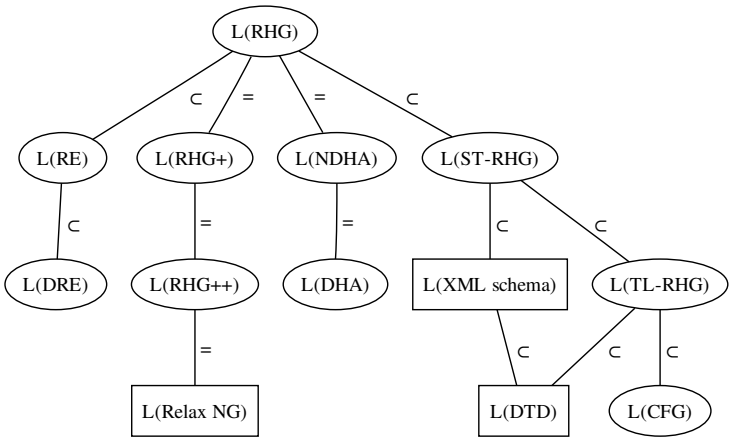
Het volgende is dus te zeggen:

$$\mathcal{L}(\text{DTD}) \subset \mathcal{L}(\text{XML schema}) \subset \mathcal{L}(\text{Relax NG})$$

Daarnaast genereren DTD en XML schema trees en Relax NG hedges. In figuur 6.1 is schematisch weergegeven hoe de verschillende XML schema's en besproken modellen zich verhouden qua uitdrukkingskracht.

Zoals te zien kunnen alle hedges welke gegenereerd kunnen worden met de besproken schema's met behulp van een (N)DHA geaccepteerd worden.

Daarnaast is Relax NG gesloten onder de vereniging, intersectie en het complemenent, waarbij DTD en XML schema slechts gesloten zijn onder de intersectie.



Figuur 6.1: Uitdrukkingskracht van XML schema's

Bijlage A

Definities

A.1 Eindige automaten

Definitie A.1.1. *Een* deterministische eindige automaat (DFA) *[8] is een 5 tupel,* $\langle Q, \Sigma, \delta, q_0, F \rangle$*, waarbij:*

1. Q *een eindige verzameling toestanden is.*
2. Σ *een eindige verzameling symbolen is.*
3. δ *een functie van* $Q \times \Sigma$ *naar* Q *is. Dit zijn de transities.*
4. q_0 *een toestand uit* Q *is, dit is de begintoestand.*
5. F *een subset van* Q *is, dit is de verzameling geaccepteerde toestanden.*

Definitie A.1.2. *Een* non-deterministische eindige automaat (NDFA) *[8] is een 5 tupel,* $\langle Q, \Sigma, \delta, q_0, F \rangle$*, waarbij:*

1. Q *een eindige verzameling toestanden is.*
2. Σ *een eindige verzameling symbolen is.*
3. δ *een functie* $Q \times (\Sigma \cup \{\epsilon\})$ *naar* 2^Q *is. Dit zijn de transities.*
4. q_0 *een toestand uit* Q *is, dit is de begintoestand.*
5. F *een subset van* Q *is, dit is de verzameling geaccepteerde toestanden.*

A.2 Context vrije grammatica

Definitie A.2.1. *Een* context vrije grammatica (CFG) *[8] is een 4 tupel,* $\langle V, \Sigma, R, S \rangle$*, waarbij:*

1. V *een eindige verzameling non terminals is.*
2. Σ *een eindige verzameling symbolen is.*
3. R *een functie van* V *naar* $2^{(V \cup \Sigma)^*}$ *zodanig dat* $\exists_{w \in (V \cup \Sigma)^*}(S, w) \in R$*. Dit zijn de transities.*
4. S *een element uit* V *is, deze is het start symbool.*

Bibliografie

[1] Anne Brüggemann-Klein and Derick Wood. Deterministic Regular Languages. In *STACS*, pages 173–184, 1992.

[2] Wim Martens, Frank Neven, Thomas Schwentick, and Geert Jan Bex. Expressiveness and complexity of XML Schema. *ACM Trans. Database Syst.*, 31(3):770–813, 2006.

[3] Makoto Murata. Hedge Automata: a Formal Model for XML Schemata. `http://horobi.com/Projects/RELAX/Archive/hedge_nice.html`, 2000.

[4] Makoto Murata. Extended path expressions for XML. In *Symposium on Principles of Database Systems*, 2001.

[5] Makoto Murata, Dongwon Lee, and Murali Mani. Taxonomy of XML Schema Languages Using Formal Language Theory. In *Extreme Markup Languages*, 2001.

[6] RELAX NG home page. `http://relaxng.org/`.

[7] Extensible Markup Language (XML) 1.0 (fourth edition). `http://www.w3.org/TR/REC-xml/#dt-doctype`.

[8] Wikipedia: the free encyclopedia. `http://wikipedia.org`.

[9] XML schema part 1: Structures second edition. `http://www.w3.org/TR/xmlschema-1/`.