Pattern-Based Updates for the Web
Refinement of Syntax and Semantics in XChange

Abschlussvortrag
Diplomarbeit
23.08.2007

Fatih Coskun
Betreuer: François Bry, Michael Eckert

XChange

- reaktive Sprache für das Web
- pattern-based Ansatz von Xcerpt
- basierend auf ECA-Regeln
- Bestandteile:
  - Event Queries (basierend auf Xcerpt Queryterme)
  - Web Queries (Xcerpt Queries)
  - Updates (Erweiterung von Xcerpt)

Updates Bisher

- Semantik als Umschreibung in Xcerpt Transformationsregeln
- hauptsächlich für Bäume definiert
- Dokumentation nur in Prototyp-Implementierung gegeben
- Nachteile
  - (derzeit) direkt als Transformation implementiert
  - Umschreibung nicht trivial
  - Semantik nicht leicht zu verstehen

Beitrag dieser Arbeit

- Präzises Sprachdesign für XchangeUp
- Präzise Definition der XChangeUp Semantik

- Außerdem:
  - Untersuchung von Related Work
  - Usecases für XChangeUp
  - Vergleich von XChangeUp mit anderen Update Sprachen

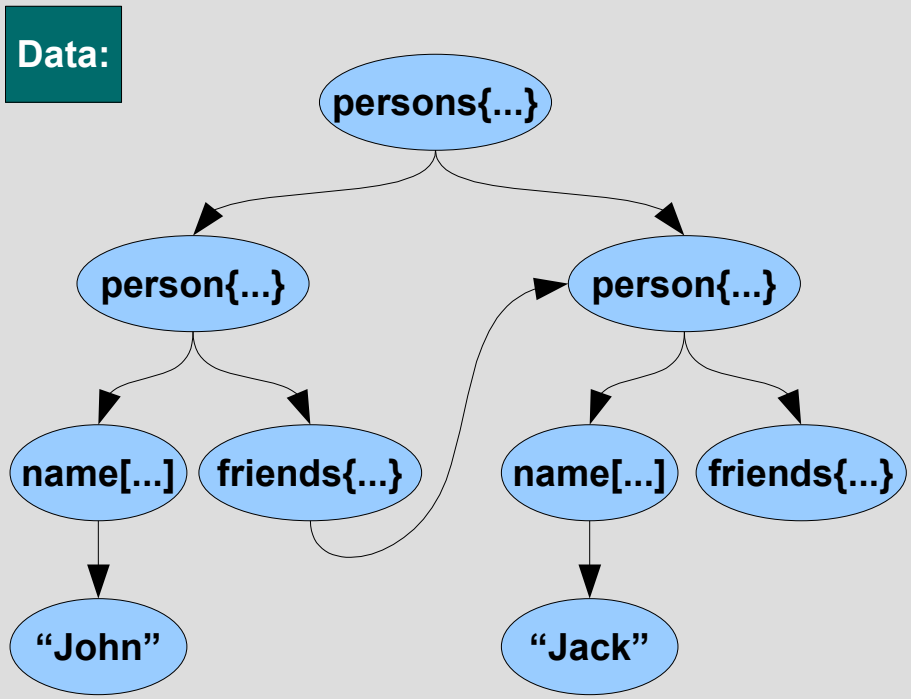
XChangeUp Sprachdesign

- Query-Teil (optional):
  - mittels Xcerpt
  - liefert Variablenbindungen (Substitutionsmengen)
- Update-Teil:
  - Update Terme: mit Update Spezifikationen angereicherte Queries
  - Insert, Delete, Replace, ...

```
FROM

    <Query-Teil>

UPDATE

    <Update-Teil>

END
```

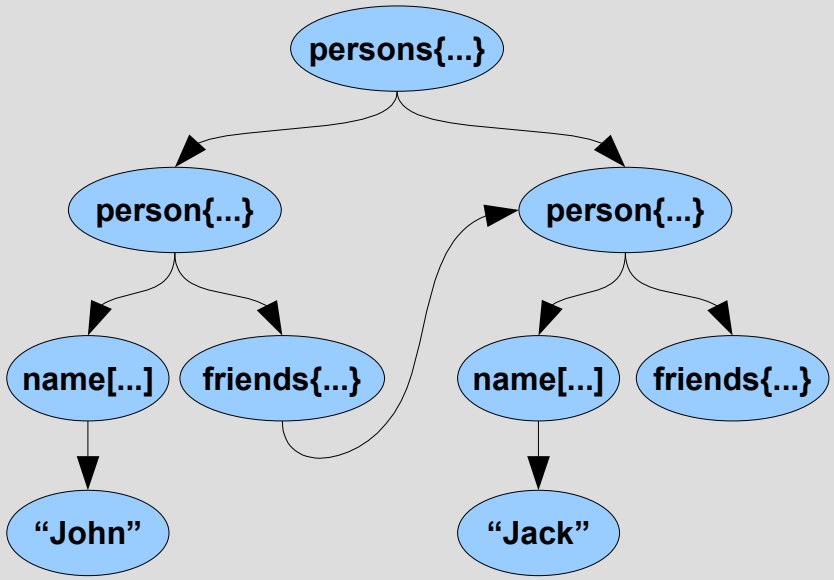
Beispiel: Delete

```
UPDATE
  in {
    resource ["file:Data"],

    persons {{
      person {{
        name ["John"],

        friends {{
          delete person {{
              name ["Jack"]
          }}
        }}
      }}
    }}
  }
END
```



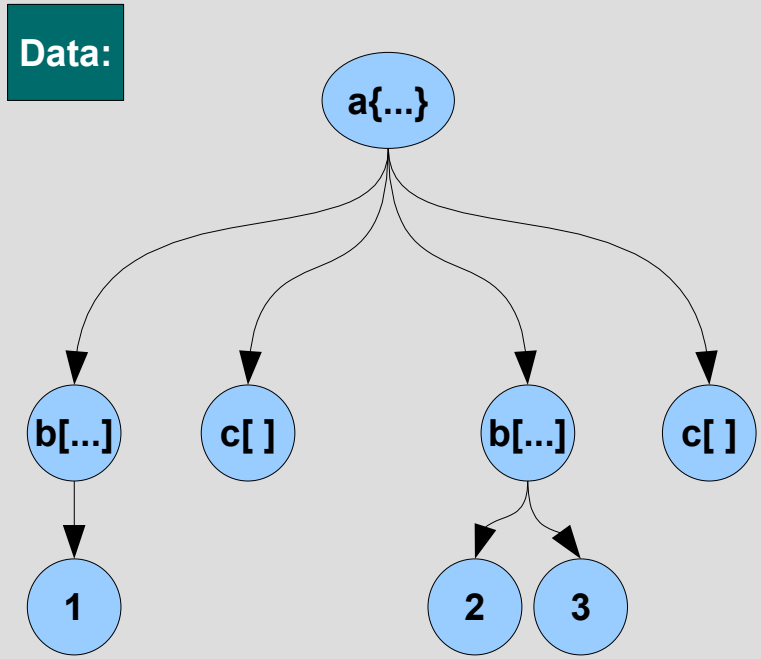
Deletion Specification

- Besteht aus einem Query Term ***Q***
- Was passiert, wenn ***Q*** nicht matched?
- Kanten löschen oder Knoten löschen?
- Welche *targets* werden gelöscht (falls mehrere möglich)?
- Wieviel wird von einem *target* gelöscht?



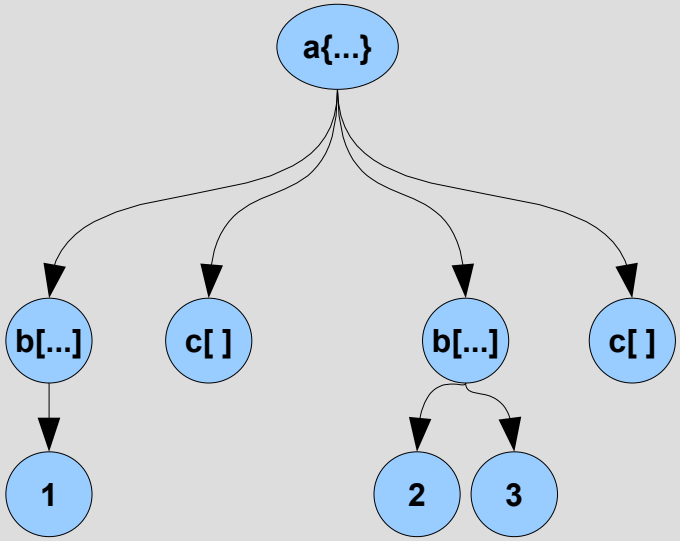
Beispiel: Replace

```
UPDATE
   in {
      resource [ "file:Data" ],
      a {{
         b[[var X]] replace by d[var X]
      }}
   }
END
```



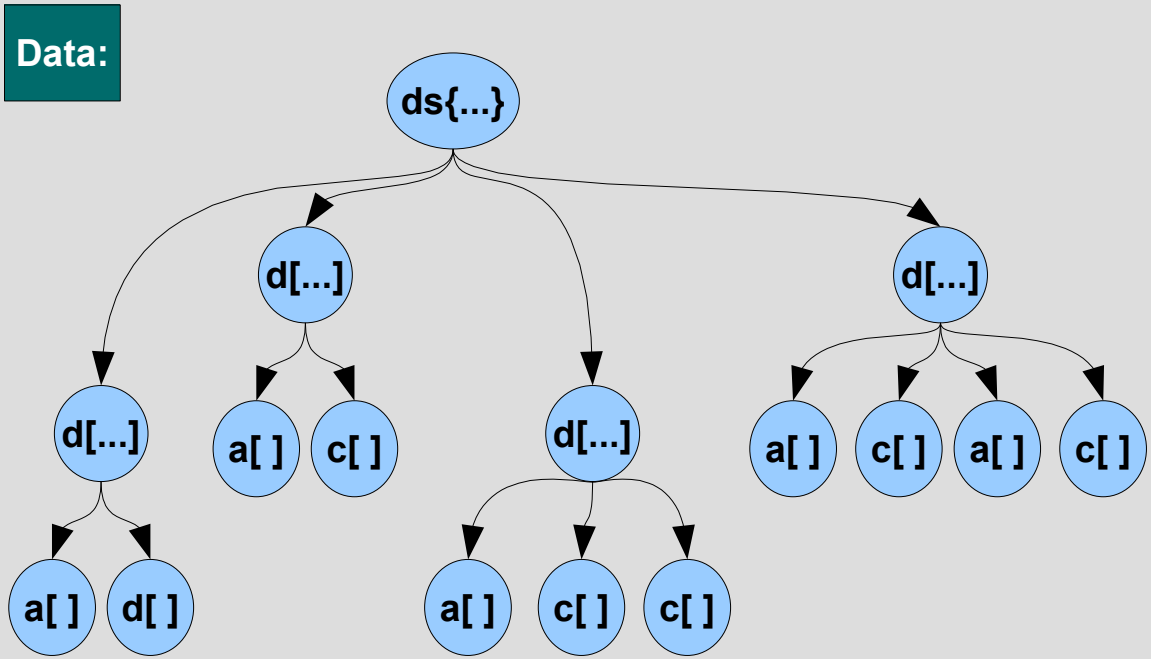
Replacement Specification

- Besteht aus einem Query Term ***Q*** und einem Construct Term ***C***
- Was passiert, wenn ***Q*** nicht matched?
- Welche *targets* werden ersetzt (falls mehrere möglich)?
- Wieviel wird von einem *target* ersetzt?
- Mehrere mögliche Variablenbindungen?
- Gruppierungen in ***C*** betrachten welche Bindungen?



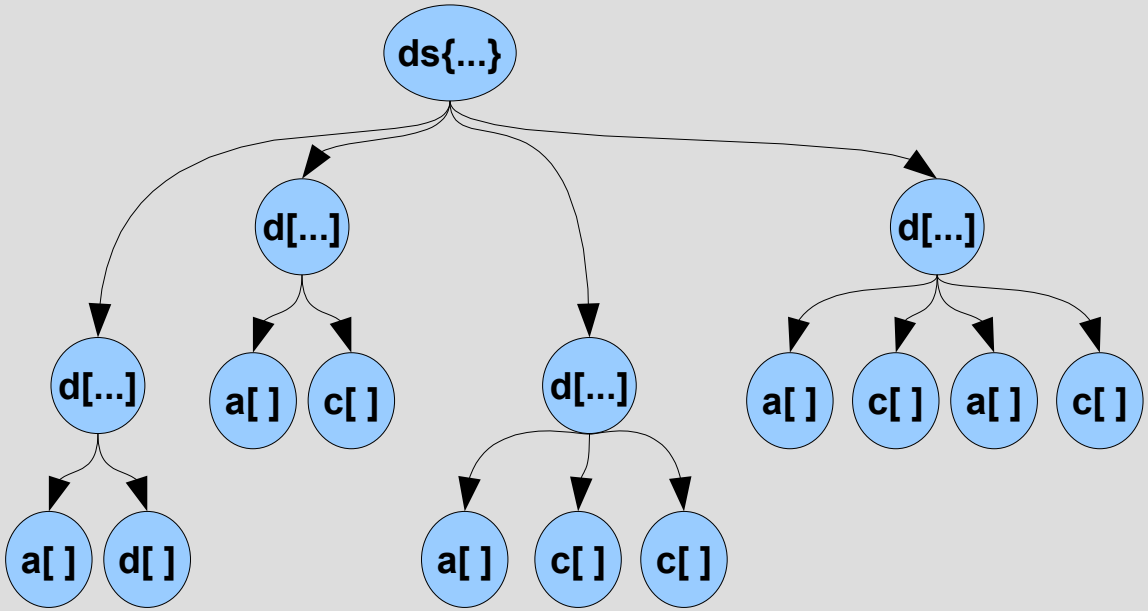
Beispiel: Insert

```
UPDATE
  in {
    resource ["file:Data"],

    desc d [[
      a,
      insert b,
      c
    ]]
  }
END
```



Insertion Specification

- Besteht aus einem Construct Term ***C***
- Mehrere mögliche *target* Positionen?
- Mehrere Einfügungen an der selben Position?
- Mehrere mögliche Variablenbindungen?
- Gruppierungen in ***C*** betrachten welche Bindungen?



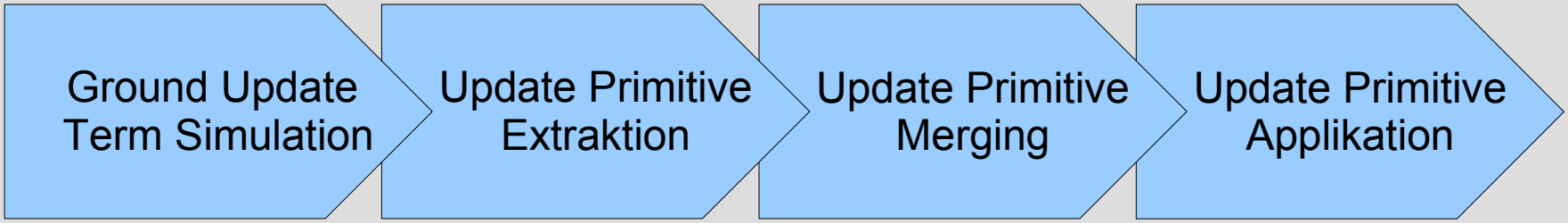
XChangeUp Semantik

Grundprinzip der Semantik:

- Rückführung der Updates auf Anfragen
- Ergebnis der “Anfragen” sind Update Primitive (= durchzuführende Updateoperationen)

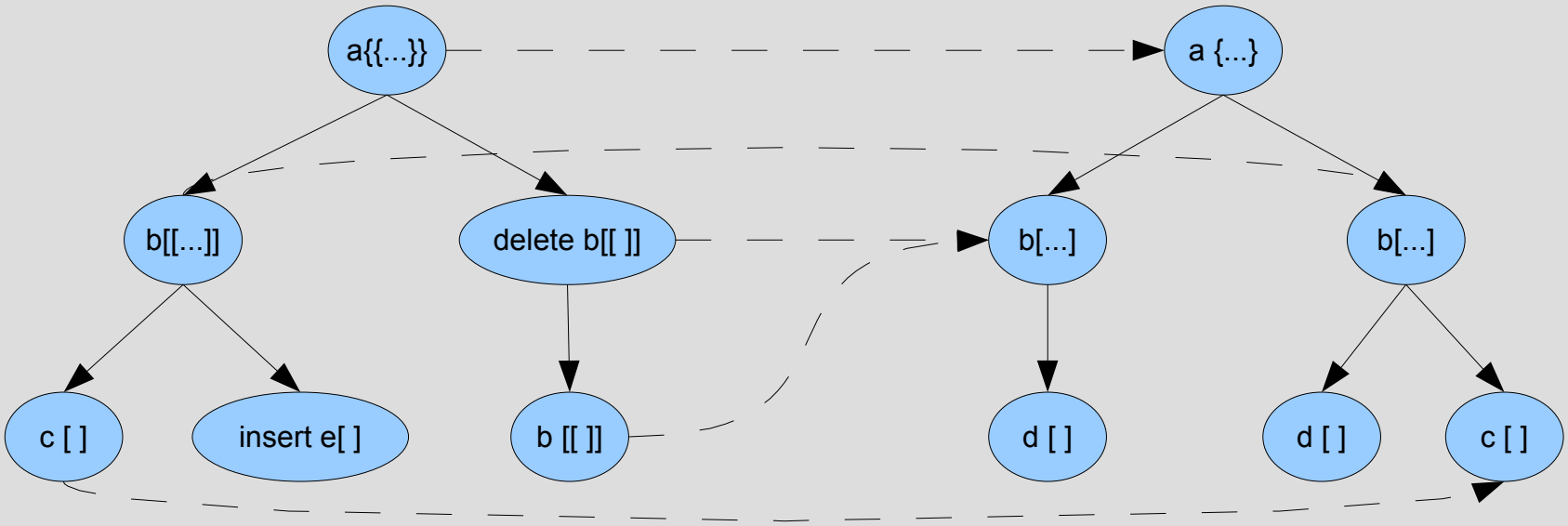
Vorteil:

- deklarativ
- Effizienter Update in Datenbanken möglich



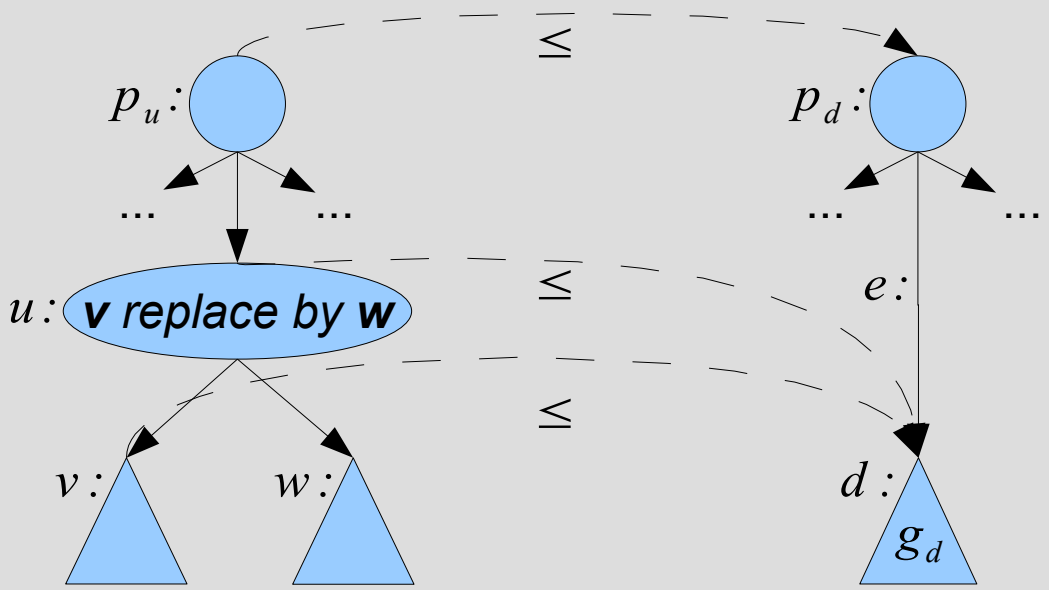
Ground Update Term Simulation

- Erweiterung der ground query term simulation auf update terme
- Terme der Form ***delete q*** werden abgebildet
- Terme der Form ***q replace by c*** werden abgebildet (auf q-Anteil)
- Terme der Form ***insert c*** werden nicht abgebildet (Einfügeposition ergibt sich aus der Umgebung)



Update Primitive und deren Extraktion

- Kapseln Information über ein Update (Position, Construct Term, Substitutionen); entsprechen Basisoperationen
- Pro Simulation wird für eine Update Spezifikation genau ein Update Primitive extrahiert
- Alle möglichen Simulationen müssen betrachtet werden



$$UP_{rep}(G_u, G_d, \leq, \sigma) = \{ replaceNode_p(d, w, \{\sigma\}) \mid u \leq d, \; u \text{ hat die Form } v \; replace\,by \; w \}$$

Update Primitive Merging (1)

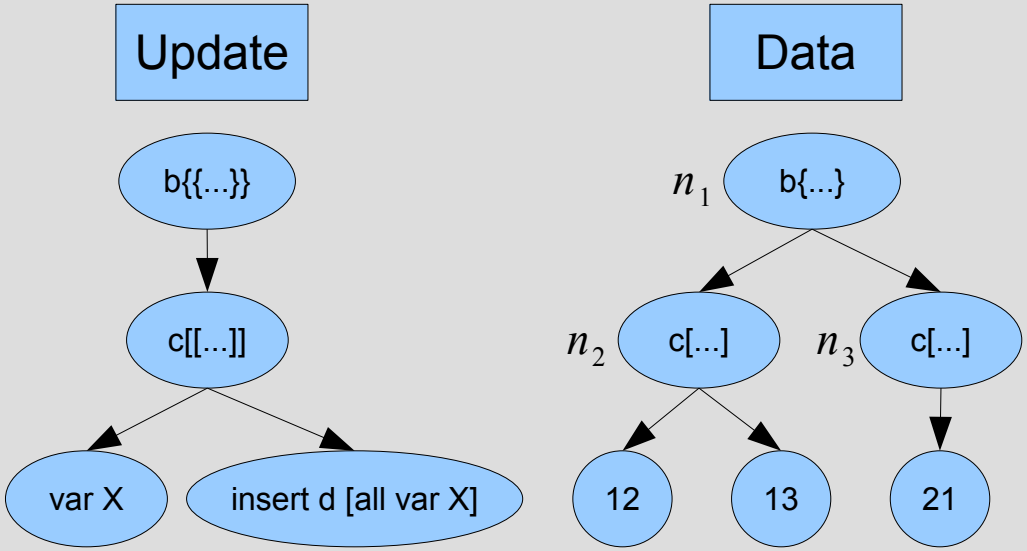
- *Union* von Update Primitiven zum Vereinigen von Substitutionsmengen
- Nötig bei allen *Insertion* und *Replacement* Primitiven

Extrahierte Primitive :

$$insertInto_P(n_2, d[all\ var\ X], \{\{X=12\}\})$$
$$insertInto_P(n_2, d[all\ var\ X], \{\{X=13\}\})$$
$$insertInto_P(n_3, d[all\ var\ X], \{\{X=21\}\})$$

Vereinigte Primitive :

$$insertInto_P(n_2, d[all\ var\ X], \{\{X=12\}, \{X=13\}\})$$
$$insertInto_P(n_3, d[all\ var\ X], \{\{X=21\}\})$$



Update Primitive Merging (2)

- *Subsumption* von Update Primitiven zum Finden der *spezifischten* Position
- Nur nötig bei *insertBetween* Primitiven (da keine *feste* Einfügeposition existiert)

Extrahierte Primitive:

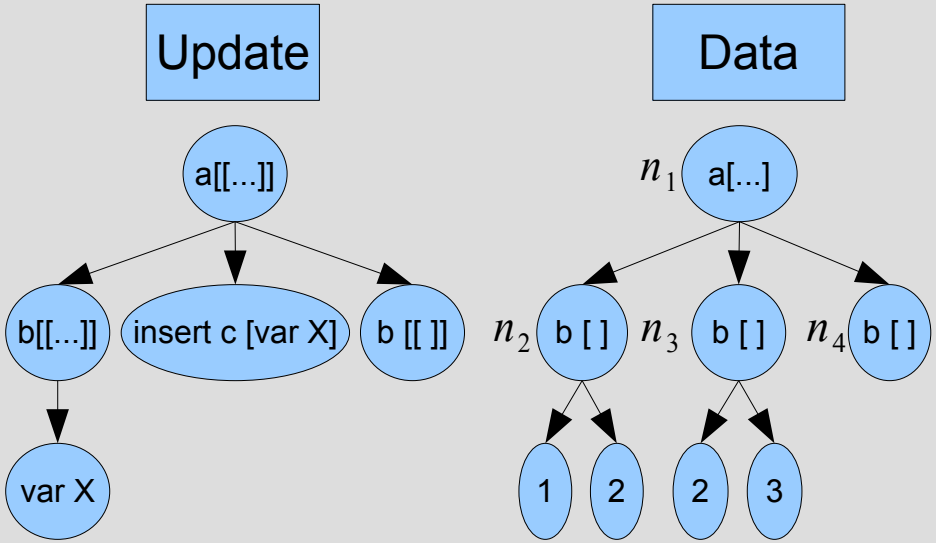
$$insertBetween_P(n_2, n_3, c[var\ X], \{\{X=1\}\})$$
$$insertBetween_P(n_2, n_3, c[var\ X], \{\{X=2\}\})$$
$$insertBetween_P(n_2, n_4, c[var\ X], \{\{X=1\}\})$$
$$insertBetween_P(n_2, n_4, c[var\ X], \{\{X=2\}\})$$
$$\ldots$$

Vereinigte Primitive:

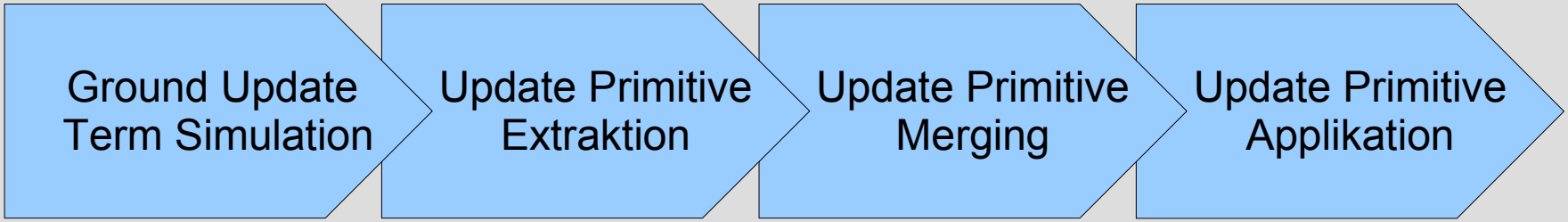
$$insertBetween_P(n_2, n_3, c[var\ X], \{\{X=1\}, \{X=2\}\})$$
$$insertBetween_P(n_2, n_4, c[var\ X], \{\{X=1\}, \{X=2\}\})$$
$$insertBetween_P(n_3, n_4, c[var\ X], \{\{X=2\}, \{X=3\}\})$$

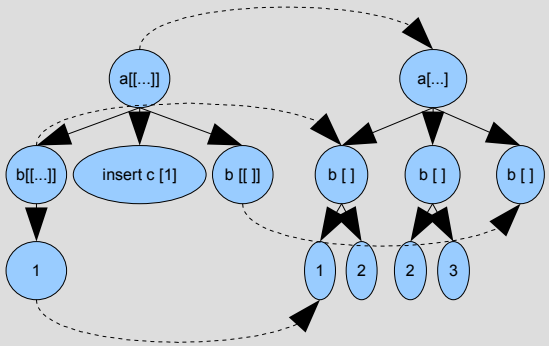
Subsumierte Primitive:

$$insertBetween_P(n_2, n_3, c[var\ X], \{\{X=1\}, \{X=2\}\})$$
$$insertBetween_P(n_3, n_4, c[var\ X], \{\{X=3\}\})$$

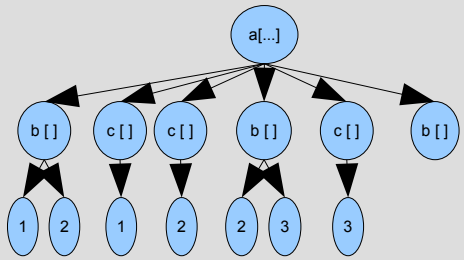


XChangeUp Semantik





$$insertBetween_P(n_2, n_3, c[var\ X], \{\{X=1\}, \{X=2\}\})$$
$$insertBetween_P(n_3, n_4, c[var\ X], \{\{X=3\}\})$$



Vergleich mit anderen Update Sprachen

	Separation of Query and Update	Selection Approach	Identification of Update Position
XQuery Update Facility	No separation (nesting)	Path-based	Identity variable
XML-RL Update Language	Strict separation	Pattern-based	Identity variable
XChangeUp	*Semi* separation	Pattern-based	Pattern-based

XChangeUp

```
persons [[
  person [[
    name [ var N ]
    skill [ var S ]
  ]],
  insert skill [
    name [ var S ],
    all person [ var N ]
  ]
]]
```

XQuery Update Facility

```
for $S in //skill/child::text()
do insert
    <skill>
      <name>{$S}</name>
      for $P in //person
      WHERE $P/skill/ = $S
      RETURN
        <person>
          {$P/name/child::text()}
        </person>
    </skill>
into
    /persons
```

XML-RL Update

```
query
  persons => [
    person => [
      name => $N,
      skill => $S
    ]
  ]
update
  insert skill => [
    name => $S,
    person => {$N}
  ]
  into /persons => $
```

Zusammenfassung und Ausblick

- Zusammenfassung:
  - Design und Definition der Sprache
  - Definition der Semantik
  - Untersuchung und Vergleich von Update Sprachen
- Ausblick:
  - Definition von Syntax und Semantik von weiteren Konstrukten
  - Erweiterung der Simulations Unifikation
  - Implementierung von XChangeUp