Automata for Avoiding Unnecessary Ordering Operations in XPath Evaluation Plans

Mary Fernández
Jan Hidders
Philippe Michiels
Roel Vercammen[1]
Jérôme Siméon[2]

July 15, 2004

---

[1] Philippe Michiels and Roel Vercammen are supported by IWT – Institute for the Encouragement of Innovation by Science and Technology Flanders, grant numbers 31016 and 31581.

[2] Jérôme Siméon completed part of this work at Lucent Technologies – Bell Labs.

Abstract

XPath 2.0 path expressions can observe and preserve the document order and identity of XML values in a document. In particular, their semantics requires that the complete result and the result of each individual step in a path expression be in document order and duplicate-free. Implementations of this semantics often guarantee correctness by inserting explicit operations that sort and remove duplicates after each step. Such operations, however, can be redundant, because an intermediate result may already be sorted and/or duplicate-free. This work presents a sound and complete set of inference rules that decide whether each step in a path expression always yields a result in document order and with no duplicates. The inference rules are implemented by an efficient, automaton-based algorithm. Experimental results show that the algorithm detects and eliminates all redundant sorting and duplicate elimination operators, and is effective on most common path expressions.

Contents

Chapter 1

Introduction

XML is an inherently ordered data format, that is, the relative order of elements, comments, processing instructions, and text in an XML document is significant. This makes XML an ideal format for data in which order is semantically significant. The order property is closely related to uniqueness: Two XML elements that are structurally identical can be distinguished by their location in a document, that is, they have distinct identities even though their internal structures are identical. Like the order property, unique identity can convey semantics.

Text-rich XML documents, such as manuscripts and transcripts, typically depend on order. For example, the order of witnesses and their statements in the transcript of a trial conveys a temporal order. If the order of witnesses' remarks changes, the meaning of the transcript is changed. Order can also be significant in data that is not text rich. For example, the order of entries in a Web-server cache can convey the order in which the cache was populated. For example, the same witness making the same remark three consecutive times in his testimony is significant. Removing the otherwise equivalent remarks changes the meaning of the testimony. Similarly, multiple structurally identical entries in a Web-server log may indicate an attempted security attack.

The XML data model that underlies the XML query languages XPath 2.0[2], XSLT 2.0[15], and XQuery 1.0[3], models both document order and node identity. Document order is a total ordering of nodes in an XML document and is the order returned by a preorder traversal of the XML document. All three query languages provide features that can observe and preserve the document order and identity of XML values in a document. XPath path expressions navigate through the axes of a document (e.g., `child`, `ancestor`, `following-sibling`), select nodes in an axis based on their name, type, or relative order with respect to other nodes, and return a sequence of selected nodes in document order and with no duplicates. XPath 2.0 is a proper sub-language of XQuery 1.0, therefore, XQuery 1.0 is at least as expressive. In addition, XQuery has several operations and functions (e.g., the `<<` and `>>` operators, and `union` and `except` functions) that depend on or yield duplicate-free node sequences in document order. Therefore, any serious implementation of XPath, XSLT, or XQuery must support the ordered data model and correctly implement the semantics of expressions that observe and preserve order. Path expressions, in particular, are integral to all three languages, and therefore, implementing them correctly, completely, and efficiently is important.

The complete formal definition of path expressions is in the XQuery 1.0 Formal Semantics [5]. The semantics is expressed through *normalization rules*, which translate users' expressions into a smaller core language. The semantics of a path expression corresponds to a top-down evaluation plan and requires that the complete result and the result of each individual step be in document order and duplicate-free. This semantics guarantees that steps that depend on order, e.g., positional predicates, always yield correct results. Because any step may yield a sequence of nodes without these properties, the semantics is enforced by inserting explicit operations, called `ddo` operations (for `distinct-docorder`), that sort and remove duplicates after each step. In many cases, these operations are redundant, because the result after certain steps is always sorted and/or duplicate-

```
<!ELEMENT surgery procedure*>
<!ELEMENT procedure
          (anesthesia | incision | subproc | #PCDATA)*>
<!ELEMENT subproc   (anesthesia | incision | #PCDATA)*>
<!ELEMENT anesthesia        #PCDATA>
<!ELEMENT incision          #PCDATA>
```

Figure 1.1: DTD of surgical procedures

free. Eliminating redundant operators yields path expressions that are semantically correct, but easier to evaluate efficiently by enabling, for example, non-blocking, pipelined operators.

The main contribution of this work is a comprehensive technique for detecting and eliminating redundant sorting and duplicate-elimination operators in XPath 2.0. In particular, this includes:

- A sound and complete set of inference rules that deduce whether each step in a path expression always yields a result in document order and with no duplicates, independent of the document to which the path expression is applied;
- An efficient, automaton-based algorithm that implements the inference rules during static analysis of a query; and
- Experimental results that show the algorithm detects and eliminates all redundant sort and duplicate-elimination operators and is very effective on typical path expressions.

Because our goal is to support the complete XPath 2.0 language, we consider all thirteen axes and boolean and positional predicate expressions. Our only assumption is that a step expression (i.e., an *axis*, *node-test* pair) applied to *one* node always yields a duplicate-free node *sequence* in document order. Note, however, that a step expression applied to a sequence of nodes may result in a node sequence out of document order or with duplicates.

The algorithm is implemented in the Galax XQuery engine [6] as a logical rewriting of XQuery core expressions. To support rewriting of path expressions in any XQuery expression, the proposed algorithm requires *weak typing*, an inexpensive and easy-to-implement form of static typing, making the algorithm applicable in other XQuery implementations.

This report is organized as follows. In the remainder of this chapter we define XPath expressions, their semantics and their evaluation plans. Here we also introduce the different properties of path expressions that we will reason about with the inference rules. In Chapter 2 we discuss the problem for so-called tidy evaluation plans that make sure that after each step the result is sorted and without duplicates. In Chapter 4 we look at how this work can be applied to improve the performance of the Galax XQuery engine. In Chapter 5, we relate the `DDO` optimization to other interacting optimizations and in Chapter 6 we put our techniques to the test with Galax. Finally, in Chapter 7 we discuss related work and the obtained results.

1.1 Motivating Examples

We begin with several path expressions that illustrate the difficulty of detecting whether the final or intermediate result of a path expression is in document order and duplicate-free.

The example expressions are applied to a document containing transcripts of surgical procedures, which conform to the DTD in Figure 1.1. Each procedure contains a sequence of steps (e.g., administer anesthesia, make incision, perform subprocedure) interleaved with text. Figure 1.2 depicts a tree representation of such a document containing two procedures (p), each containing anesthesia (a), incision (i), and sub-procedure (s) elements. The integer subscripts denote document order. In example expressions, an absent axis denotes `child::`, and all other axis names are abbreviated (e.g., `descendant::` becomes `desc::`). Our primary goal is to decide whether the result of each step in a path expression is always in document order (i.e., has the *ord* property) and/or never contains duplicates (i.e., has the *nodup* property).

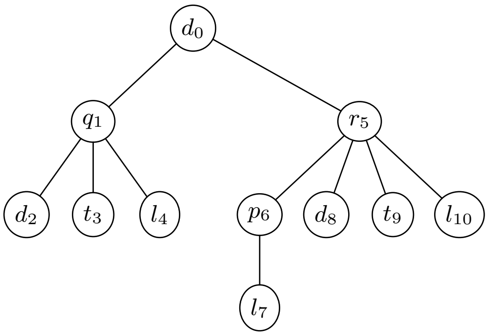


Figure 1.2: A tree representation of a surgical procedure

The first example return all incisions contained within procedures:

```
$sur/procedure/desc::incision
```

In this example the `$sur` variable is bound to node d_0 of Figure 1.2. By simple inspection, we can infer the first `procedure` step is *ord* and *nodup*, and because the descendants of the `procedure` nodes are unrelated, we can infer that `desc::incision` is also *ord* and *nodup*.

The above expression is normalized into the following core expression, which makes explicit the semantics that the complete result and the result of each individual step be in document order and duplicate-free. Each step is evaluated with respect to an implicit *context node*, which is bound to the variable `$fs:dot`. The `fs:distinct-docorder` function sorts its input in document order and removes duplicates.[1]

```
fs:distinct-docorder(
  for $fs:dot in fs:distinct-docorder(
                   for $fs:dot in $sur
                   return child::procedure)
  return descendant::incision)
```

Our algorithm infers that each step in the above core expression is always in document order and duplicate-free and therefore can be simplified into the equivalent core expression:

```
for $fs:dot in
  for $fs:dot in $sur return child::procedure
return descendant::incision
```

Deducing the *ord* and *nodup* properties for path expressions involving both forward and reverse axes is more difficult. The following expression uses the parent axis to return elements that directly contain incisions contained within some procedure:

```
$sur/procedure/desc::incision/..
```

As before, the first two steps are *ord* and *nodup* but the parent step may yield a sequence that is neither *ord* nor *nodup*. For example, the parent axis applied to the sequence of incisions $\langle i_3, i_5, i_9, i_{11} \rangle$ is the sequence $\langle p_1, p_1, s_8, p_7 \rangle$. Our algorithm can simplify the normalized core expression, eliminating unnecessary `fs:distinct-docorder` operations, while preserving those that are necessary.

Sometimes the intermediate result of a step may be ordered but contain duplicates, or vice versa. For example, replacing the `desc::incision` axis by the `child::incision` axis in the previous expression yields a result that is always in document order but that may contain duplicates:

```
$sur/procedure/incision/..
```

---

[1] The `fs` namespace denotes "Formal Semantics".

For example, the parent axis applied to the sequence of incisions $\langle i_3, i_5, i_{11}\rangle$ is the sequence $\langle p_1, p_1, p_7\rangle$.

Our algorithm infers when a path expression is ordered but may contain duplicates and replaces the `fs:distinct-docorder` operation by the more efficient `fs:disinct` operation, which takes an ordered sequence and in linear time removes duplicates:

```
fs:distinct(
  for $fs:dot in
    for $fs:dot in
        for $fs:dot in $sur
        return child::procedure
    return child::incision
  return parent::node())
```

Much existing work on XPath semantics considers a small subset of the language, ignoring positional predicates and backward and sibling axes. Our experience is that many applications require all of XPath, and therefore our algorithm is designed to handle the complete XPath 2.0 language. We also expect that as automatic generation of XPath expressions increases, XPath processors will be required to implement them correctly and efficiently.

1.2 XPath

We continue with the theoretical foundation for inferencing the *ord* and *nodup* properties. We begin with a formal definition of axes and path expressions, then introduce two *evaluation plan categories* that correspond top-down evaluations of a path expression. Our goal is to decide whether `ddo` operations in the evaluation plan of a path expression can be removed without changing its semantics. We do this by deciding the properties *ord* and *nodup* after each step in a path expression.

To derive these static properties, we need to introduce several auxiliary static properties of evaluation plans and the inference rules that derive them. This yields a *sound* and *complete* set of rules for deciding the *ord* and *nodup* properties. The rules are sound, because they never derive *ord* or *nodup* for an expression that can yield a sequence that is not in order or that contains duplicates. The rules are complete, because, if they cannot derive *ord* or *nodup* for an expression, then there exists at least one evaluation that yields a sequence that is unordered or contains duplicates. Lastly, we show that these rules can be realized by an efficient decision algorithm using a deterministic automaton.

We note that in XPath 2.0, each step in a path consists of an axis and a node test followed by an optional predicate, e.g., `child::procedure/descendent::incision[1]`. Here, we ignore node tests and predicates and focus on deriving the *ord* and *nodup* properties for sequences of axes, e.g., `child::*/descendant::*`. In Chapter 4, we explain how these properties are derived for complete XPath expressions. Our formalism omits path expressions in which parentheses are used to enforce right-associativity, such as in $p_1/(p_2/p_3)$, and the `self` axis, because, without a node test, it denotes the identity function.

1.2.1 Semantics of XPath

To save space, our formalization of an XML document contains only element nodes (N) labeled with tags (T). Other nodes (text, comment, etc.) can be added easily.

Definition 1.2.1 (XML Document). *An* XML document *is a rooted ordered node-labeled tree $D = (N, \lhd, r, \lambda, \prec)$ such that $(N, \lhd)$ is a tree with root r, $\lambda : N \rightarrow T$ is a labeling of the nodes, $\lhd$ is the binary parent-child relation, and $\prec$ is the sibling-order relation that is a strict partial order over N such that for each two distinct nodes $n_1, n_2 \in N$ it holds that $n_1 \prec n_2$ or $n_1 \prec n_2$ iff they are siblings.*

Axis Name	**Axis Symbol**	**Set Semantics** $\{\![\mathrm{Axis}]\!\}_D$
`child`	$\downarrow$	$\lhd$
`parent`	$\uparrow$	$\rhd$
`descendant`	$\downarrow^+$	$\lhd^+$
`ancestor`	$\uparrow^+$	$\rhd^+$
`descendant-or-self`	$\downarrow^*$	$\lhd^*$
`ancestor-or-self`	$\uparrow^*$	$\rhd^*$
`following`	$\twoheadrightarrow$	$\rhd^* \circ \prec \circ \lhd^*$
`preceding`	$\twoheadleftarrow$	$\rhd^* \circ \succ \circ \lhd^*$
`following-sibling`	$\dot{\twoheadrightarrow}$	$\prec$
`preceding-sibling`	$\dot{\twoheadleftarrow}$	$\succ$

Table 1.1: Axis names, symbols, and set semantics

Definition 1.2.2 (Document Order). *Given an XML document* $D = (N, \lhd, r, \lambda, \prec)$ *we define the* document order in D, $\ll_D$, *as the strict total order over* N *that orders the nodes as encountered in a pre-order tree-walk, i.e., the unique strict total order that is a superset of both* $\lhd$ *and* $\prec$ *and for which it holds for all* $n_1, n_2, n_3 \in N$ *that if* $n_1 \lhd^+ n_2$ *and* $n_1 \prec n_3$ *then* $n_2 \ll_D n_3$.

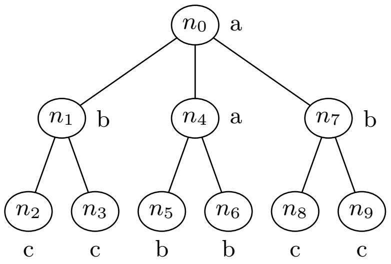


Figure 1.3: An example of an XML document

The relations $\lhd^+$ and $\lhd^*$ denote the transitive closure and the reflexive and transitive closure of $\lhd$, respectively. The reverses of $\prec$, $\lhd$, $\lhd^+$ and $\lhd^*$ are denoted by $\succ$, $\rhd$, $\rhd^+$ and $\rhd^*$, respectively.

We first define a *set semantics* and a *sequence semantics* for each axis in terms of the above relations on nodes, then define the set and sequence semantics for a path expression, which is a finite sequence of axes.

Definition 1.2.3 (Axes). *The* set of axes A *is defined as* $\{\uparrow, \downarrow, \uparrow^+, \downarrow^+, \uparrow^*, \downarrow^*, \twoheadleftarrow, \twoheadrightarrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}$ *where these symbols represent the XPath axes as given in Table 1.1. The concise notation in Table 1.1 extends the notation in [1].*

Definition 1.2.4 (Axes Semantics). *The* set semantics *of an axis* a *on a document* $D = (N, \lhd, r, \lambda, \prec)$ *is a binary relation* $\{\![a]\!\}_D \subseteq N \times N$ *and is defined by the third column of Table 1.1. For example, the semantics of the* ***`following`*** *axis is defined such that it contains the pair* (n_1, n_2) *iff* n_2 *is the descendant (or the node itself) of a node that is a following sibling of an ancestor (or the node itself) of* n_1[2].

The sequence semantics *of an axis* a *on a document* $D = (N, \lhd, r, \lambda, \prec)$ *is a function* $[\![a]\!]_D : N \to \mathcal{S}(N)$ *where* $\mathcal{S}(N)$ *denotes the set of finite sequences over* N *such that* $[\![a]\!]_D(n)$ *is the sequence that is obtained by sorting the set* $\{n' | (n, n') \in \{\![a]\!\}_D\}$ *with* $\prec$, *the document order of* D.

We overload the last notation and define a function $[\![a]\!]_D : \mathcal{S}(N) \to \mathcal{S}(N)$ *such that lin ebreak* $[\![a]\!]_D(\langle n_1, \ldots, n_k\rangle) = [\![a]\!]_D(n_1) \cdot \ldots \cdot [\![a]\!]_D(n_k)$ *where* $\cdot$ *denotes sequence concatenation.*

---

[2] The composition of two binary relations R and S is $R \circ S = \{(n_1, n_3) | (n_1, n_2) \in S, (n_2, n_3) \in R\}$

Definition 1.2.5 (Path Expression). *A* path expression *is a non-empty finite sequence of axes, denoted as* $a_1/\ldots/a_k$.

Finally, we define the semantics of a path expression, which coincides with the semantics of path expressions in the XQuery formal semantics.

Definition 1.2.6 (Semantics of Path Expressions). *The* semantics of a path expression *$p = a_1/\ldots/a_n$ on a document $D = (N, \lhd, r, \lambda, \prec)$, is the function $[\![p]\!]_D : N \rightarrow \mathcal{S}(N)$ such that the result of $[\![p]\!]_D(n)$ is the sequence obtained when sorting the set $\{n'|(n,n') \in \{\![a_n]\!\} \circ \ldots \circ \{\![a_1]\!\}\}$ with $\ll_D$, the document order of D.*

In other words, the result of a path expression is the set of nodes that is the result of the composition of the semantics of each step in the expression, and this set is returned as a sequence of nodes that is sorted by document order.

1.2.2 Evaluation Plans for Path Expressions

The above semantics specifies what a path expression means, but not how to evaluate it. Therefore we introduce a notion of an *an evaluation plan* which allows us to reason about the necessity of sorting and duplicate-elimination operations. Informally such an evaluation plan consist of a sequence of axis symbols and the symbols σ and δ that indicate the sorting and duplicate elimination operation, respectively. Such a sequence is denoted as for example $\downarrow^*; \downarrow; \sigma; \delta; \uparrow; \delta$. The interpretation of such a sequence is that the end-result is computed in a step-by-step fashion, i.e., the operation that corresponds to each step is applied to the result of the previous step. Here the operation that corresponds to an axis symbol a is the function $[\![a]\!]_D$ over sequences of nodes, i.e., we iterate over the input sequence, apply the axis to each node and concatenate all the results.

The evaluation plans are abstractions from the core algebra expressions to which path expressions are normalized. For example the evaluation plan $\downarrow^*; \downarrow; \sigma; \delta; \uparrow; \delta$ corresponds with:

```
fs:distinct(
  for $fs:dot in fs:distinct-docorder(
    for $fs:dot in descendent-or-self::*
    return child::*)
  return parent::*)
```

Therefore we can derive properties of these concrete evaluation plans in the core algebra by reasoning about the abstract evaluation plans.

We now proceed with the formal definition. For this we introduce the symbols σ and δ, which denote a sorting operator and a duplicate elimination operator, respectively. We define their semantics with the function $[\![\sigma]\!]_D : \mathcal{S}(N) \rightarrow \mathcal{S}(N)$, which sorts a sequence of nodes with $\prec$, and the partial function $[\![\delta]\!]_D : \mathcal{S}(N) \rightarrow \mathcal{S}(N)$, which takes a sorted sequence of nodes and removes duplicates.

Definition 1.2.7 (Evaluation Plan). *An* evaluation plan *is a non-empty sequence $q = s_1; \ldots; s_k$ of axis symbols, σ and δ where (1) the first element is an axis symbol and (2) between two axis symbols and after the last axis symbol we either find the sequence $\sigma; \delta$, the sequence σ, the sequence δ or the empty sequence. Given a document D, the* semantics of an evaluation plan *$q = s_1; \ldots; s_k$, is a partial*[3] *function $[\![q]\!]_D : N \rightarrow \mathcal{S}(N)$ such that $[\![q]\!]_D(n) = F(\langle n \rangle)$ where $F = [\![s_k]\!]_D \circ \ldots \circ [\![s_1]\!]_D$.*

With every evaluation plan corresponds a path expression that it is supposed to implement. This path expression consists of the axes as encountered in the evaluation plan. For example, the evaluation plan $\downarrow^*; \downarrow; \sigma; \delta; \uparrow; \delta$ implements $\downarrow^*/\downarrow/\uparrow$. Given an evaluation plan q we will write the corresponding path expression as $P(q)$. Furthermore, we say that an evaluation plan q is *correct* if it holds for all XML documents D that $[\![q]\!]_D = [\![P(q)]\!]_D$. For example, the evaluation plan $\downarrow; \uparrow; \delta$ is correct, because the result after $\uparrow$ will always be a sequence of zero or more times the same node,

---

[3] The semantics of an evaluation plan is a partial function, because δ assumes an ordered input sequence.

but $\downarrow; \uparrow$ is not correct. If an evaluation plan is correct except that it may still produce duplicates then we call it *correct up to duplicates* and if it is correct except that the result may not be sorted then we call it *correct up to ordering*. We also introduce a stronger form of correctness which we call *step correctness* an which requires that in an evaluation plan after each evaluation of an axis and any following σ and δ operations the result is always sorted and without duplicates.

Next to the semantical categories for evaluation plans defined above we also distinguish two other syntactical categories. The first is the category of *sloppy evaluation plans* which are evaluation plans that only consist of axes. The second category are the *tidy evaluation plans* which are evaluation plans that consist of zero or more times a sequence of the form $a; \sigma; \delta$, with a an axis symbol, followed by a final axis symbol. For example, for the path expression $\downarrow^*/\downarrow/\uparrow$ there is a sloppy evaluation plan $\downarrow^*; \downarrow; \uparrow$ and a tidy evaluation plan $\downarrow^*; \sigma; \delta; \downarrow; \sigma; \delta; \uparrow$. Note that both sloppy and tidy evaluation plans may be incorrect, as is illustrated by the previous example, but both can be made correct by extending them at the end with the sequence $\sigma; \delta$.

The main subject of this report will be the problem of deciding whether sloppy evaluation plans and tidy evaluation plans are correct up to duplicates and/or correct up to ordering. Depending upon the implementation strategy that is chosen this information can be used in different ways. If the implementation strategy consists of the sloppy evaluation plan followed by $\sigma; \delta$ then we can decide whether these final two steps are indeed necessary. It is easy to see that a sloppy evaluation plan produces the same result as a depth first evaluation strategy that does not materialize the intermediate node sequences but iterates over them and for each evaluates the remainder of the path expression. Therefore, we can also in this case use this information to decide whether the final σ and δ steps are really necessary.

The two previous strategies have the well-known problem that intermediate results may contain duplicates and therefore lead to duplicate computations and even an unnecessary exponential growth of the execution time in the size of the path expression [9]. This can be solved by an implementation strategy that consists of the tidy evaluation plan followed by $\sigma; \delta$. Here we can obviously use the information whether the tidy evaluation plan itself is already correct or not, to decide whether to remove the final $\sigma; \delta$. Moreover, we can use the same algorithm to decide for each of the intermediate σ's and δ's if they are redundant or not. As a consequence it allows us to decide exactly which σ's and δ's can be removed while remaining step correct. This approach is especially interesting for implementations of XQuery such as Galax that follow the formal semantics of XQuery closely. This is because a straightforward interpretation of the formal semantics lead to an implementation strategy that corresponds to the tidy evaluation plan.

1.2.3 Evaluation Plan Properties

As explained in the previous section our goal is to decide whether it holds for a certain evaluation plan whether for all XML documents and nodes in that document the result of the evaluation plan applied to that node in that document is without duplicates and sorted in document order. We can rephrase this problem as a decision problem where the following two properties have to be decided for an evaluation plan.

Definition 1.2.8 (The *ord* and *nodup* Properties). *For a path evaluation plan q we define the following properties:*

ord (Ordered) *For every XML document D and node n_1 in D the list $[\![q]\!]_D(n_1)$ is sorted in the document order of D.*

nodup (No Duplicates) *For every XML document D and node n_1 in D the list $[\![q]\!]_D(n_1)$ contains no duplicates.*

The fact that a certain property π holds for a path evaluation plan q is denoted as $q : \pi$. For example, $\downarrow; \downarrow : ord$ denotes the fact that the result of the path evaluation plan $\downarrow; \downarrow$ is always sorted in document order.

In order to decide these properties we introduce several other properties that we can use to derive them.

Definition 1.2.9 (Ancillary Properties). *For evaluation plans q we define the following properties:*

- **lin** (Linear) *For every XML document D and node n_1 in D all the nodes in $[\![q]\!]_D(n_1)$ are ancestor-descendent related.*
- **unrel** (Unrelated) *For every XML document D and node n_1 in D all the nodes in $[\![q]\!]_D(n_1)$ are* not *ancestor-descendant related.*
- **nolc** (No Left Child) *For every XML document D, node n_1 in D and nodes n_2, n_3 in $[\![q]\!]_D(n_1)$ it holds that if n_2 has a sibling n_4 that is an ancestor of n_3 then n_2 is not a left sibling of n_4.*
- **norc** (No Right Child) *For every XML document D, node n_1 in D and nodes n_2, n_3 in $[\![q]\!]_D(n_1)$ it holds that if n_2 has a sibling n_4 that is an ancestor of n_3 then n_2 is not a right sibling of n_4.*
- **nsib** (n Siblings) *For any number n there is an XML document D and a node n_1 in D such that there are at least n distinct siblings in $[\![q]\!]_D(n_1)$.*
- **no2d** (No 2 Distinct Nodes) *For every XML document D and node n_1 in D there are not two distinct nodes in $[\![q]\!]_D(n_1)$.*
- **ntree** (Tree of size n) *For any number n there is an XML document D and a node n_1 in D such that there is set of nodes in $[\![q]\!]_D(n_1)$ that spans a tree in D of height n with all internal nodes having n children.*
- **nhat** (Hat of n Siblings) *For any number n there is an XML document D and a node n_1 in D such that there is a node n_2 in $[\![q]\!]_D(n_1)$ such that there is an ancestor of n_2 which has at least n distinct left siblings and n distinct right siblings in $[\![q]\!]_D(n_1)$.*

The properties defined above are all *set properties* in the sense that they only refer to the result set of the evaluation plan and do not care about the order or the multiplicity of the nodes in the result. It will be clear that this is not true for the *ord* and *nodup* properties.

The set properties are divided in *positive set properties* and *negative set properties*. The positive set properties are those set properties that forbid certain combinations of nodes in the result of the evaluation plan, such as for example the properties *no*2*d*, *lin*, *unrel*, *nolc* and *norc*. The negative properties are those set properties that require that certain combinations can occur in the result of the evaluation plan. Examples of negative properties are *nsib*, *ntree* and *nhat*.

The fact that a certain property does *not* hold for a certain evaluation plan will also be relevant for us since, for example, we want to be able to derive when *ord* holds and when it does not hold. Therefore we introduce negated versions of all properties π which will be written as $\neg\pi$. The semantics of $q : \neg\pi$ is then assumed to be that it does not holds that $q : \pi$. So, if $q : \neg ord$ then it is not true that the result of q is always sorted. Note that this does not mean the the result is always unsorted. Also note that it is easy to see that the negated version of a positive set property becomes a negative set property and vice versa.

For many properties π it holds that if for an evaluation plan q it holds that $q : \pi$ then the same property also holds for q extended with $\downarrow;\uparrow$, i.e., $q;\downarrow;\uparrow : \pi$. For example this holds for the *ord* property, but not for the *nodup* property. In fact, this often holds if we extend q with i times the $\downarrow$ axis followed by i times the $\uparrow$ axis. Therefore, we introduce indexed versions of all the properties that indicate that the original property is obtained if we apply the $\uparrow$ axis i times.

Definition 1.2.10 (Indexed evaluation plan Properties). *A property π can have indices such as in π_i, $\pi_{\leq i}$ and $\pi_{\geq i}$, which are defined as follows:*

- $q : \pi_0$ *iff* $q : \pi$
- *if* $i > 0$ *then* $q : \pi_i$ *iff* $(q;\uparrow) : \pi_{i-1}$.
- $q : \pi_{\leq i}$ *iff for all* $j \leq i$ *it holds that* $q : \pi_j$.

- $q : \pi_{\geq i}$ *iff for all* $j \geq i$ *it holds that* $q : \pi_j$.

For all properties π *we will use* π, π_0 *and* $\pi_{\leq 0}$ *as synonyms.*

It is easy to see that if π is a positive (negative) set property then so are π_i, $\pi_{\leq i}$ and $\pi_{\geq i}$. In case the property is negated *and* has an index with $\geq$ or $\leq$ then the negation is assumed to have the higher priority. So, for example, $q : \neg\pi_{\geq i}$ means that for all $j \geq i$ it holds that $q : \neg\pi_j$.

Chapter 2

Tidy Evaluation Plans

In this chapter we discuss how we can decide for *tidy evaluation plans* whether or not the result is always ordered and free from duplicates. This information can be used to remove any unnecessary occurrences of `ddo` operations from normalized path expressions. For doing this, we present a set of rules that derive the properties *ord* and *nodup* and their negations. These rules are sound for the entire XPath/XQuery syntax, meaning that our techniques can be applied in an XQuery environment, as we will show in Chapter 4. Moreover, the set of rules is complete for the XPath fragment we consider, meaning that every `ddo` operation that can be removed is identified by the presented algorithm.

2.1 Soundness Rules

In this section we present the set of *soundness rules*, that allow us to derive the *ord* and *nodup* properties for evaluation plans. The following rule, for instance, states that if the *lin* property holds for an evaluation plan, then the *norc* property also holds:

$$\frac{q : lin}{q : norc}$$

First we present the full set of rules. Afterwards, the soundness of each of the rules is proven separately for each rule.

2.1.1 Inference Rules

Rules for *ord*

$$\frac{q : no2d, nodup \qquad a \in A}{q ; a : ord} \ \textsc{ord-no2d-step}$$

$$\frac{q : norc, ord}{q ; \uparrow : ord} \ \textsc{ord-norc-prn}$$

$$\frac{\begin{array}{l} q : unrel, ord, nodup \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q ; a : ord} \ \textsc{ord-unrel-nodup-down}$$

$$\frac{q : lin, ord}{q ; \dot{\twoheadleftarrow} : ord} \ \textsc{ord-lin-prs}$$

Rules for *nodup*

$$\frac{q : \mathit{no2d}, \mathit{nodup} \qquad a \in A}{q; a : \mathit{nodup}}\ \textsc{nodup-no2d-step}$$

$$\frac{q : \mathit{nodup}}{q; \downarrow : \mathit{nodup}}\ \textsc{nodup-chl}$$

$$\frac{\begin{array}{l} q : \mathit{lin}, \mathit{nodup} \\ a \in \{\uparrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : \mathit{nodup}}\ \textsc{nodup-lin-prnsib}$$

$$\frac{\begin{array}{l} q : \mathit{unrel}, \mathit{nodup} \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q; a : \mathit{nodup}}\ \textsc{nodup-unrel-down}$$

Rules for *no2d*

$$\frac{q : \mathit{no2d}_n \qquad n \geq 0}{q : \mathit{no2d}_{n+1}}\ \textsc{no2d-up}$$

Rules for *lin*

$$\frac{q : \mathit{no2d}_n \qquad n \geq 0}{q : \mathit{lin}_n}\ \textsc{lin-no2d}$$

$$\frac{q : \mathit{lin}_n \qquad n \geq 0}{q : \mathit{lin}_{n+1}}\ \textsc{lin-up}$$

$$\frac{q : \mathit{lin}_n \qquad n \geq 1}{q; \uparrow^+ : \mathit{lin}_{n-1}}\ \textsc{lin-anc}$$

$$\frac{q : \mathit{lin}_n \qquad n \geq 0}{q; \uparrow^* : \mathit{lin}_n}\ \textsc{lin-aos}$$

Rules for *nolc*

$$\frac{q : \mathit{lin}}{q : \mathit{nolc}}\ \textsc{nolc-lin}$$

$$\frac{q : \mathit{nolc}}{q; \dot{\twoheadrightarrow} : \mathit{nolc}}\ \textsc{nolc-fls}$$

$$\frac{q : \mathit{nolc}_n, \mathit{lin}_{n+1} \qquad n \geq 0}{q; \uparrow^* : \mathit{nolc}_n}\ \textsc{nolc-lin-aos}$$

$$\frac{q : \mathit{nolc}_n, \mathit{lin}_{n+1} \qquad n \geq 1}{q; \uparrow^+ : \mathit{nolc}_{n-1}}\ \textsc{nolc-lin-anc}$$

$$\frac{q : \mathit{unrel}_1 \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : \mathit{nolc}}\ \textsc{nolc-unrel1-sib}$$

$$\frac{q : \mathit{unrel}}{q; \downarrow : \mathit{nolc}}\ \textsc{nolc-unrel-chl}$$

$$\frac{q : \mathit{no2d}_n \qquad n \geq 0}{q : \mathit{nolc}_{\geq 0}}\ \textsc{nolc-no2d}$$

Rules for *norc*

$$\frac{q : lin}{q : norc}\ \textsc{norc-lin} \qquad \frac{q : norc}{q ; \dot{\twoheadleftarrow} : norc}\ \textsc{norc-prs}$$

$$\frac{q : norc_n, lin_{n+1} \qquad n \geq 0}{q ; \uparrow^* : norc_n}\ \textsc{norc-lin-aos} \qquad \frac{q : norc_n, lin_{n+1} \qquad n \geq 1}{q ; \uparrow^+ : norc_{n-1}}\ \textsc{norc-lin-anc}$$

$$\frac{q : unrel_1 \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q ; a : norc}\ \textsc{norc-unrel1-sib} \qquad \frac{q : unrel}{q ; \downarrow : norc}\ \textsc{norc-unrel-chl}$$

$$\frac{q : no2d_n \qquad n \geq 0}{q : norc_{\geq 0}}\ \textsc{norc-no2d}$$

Rules for *unrel*

$$\frac{q : nolc}{q ; \dot{\twoheadrightarrow} : unrel}\ \textsc{unrel-nolc-fls} \qquad \frac{q : norc}{q ; \dot{\twoheadleftarrow} : unrel}\ \textsc{unrel-norc-prs}$$

$$\frac{q : unrel_n \qquad n \geq 1}{q : unrel_{n-1}}\ \textsc{unrel-down} \qquad \frac{q : no2d_n \qquad n \geq 0}{q : unrel}\ \textsc{unrel-no2d}$$

Rules for Positive Set Properties

In order to avoid having separate rules for all positive set properties with an index, we provide some general properties that apply to all positive set properties. Hence if $P \in \{no2d, lin, nolc, norc, unrel\}$ then following rules hold:

$$\frac{q : P_n \qquad n \geq 0}{q ; \downarrow : P_{n+1}}\ \textsc{p-chl} \qquad \frac{q : P_n \qquad n \geq 1}{q ; \uparrow : P_{n-1}}\ \textsc{p-prn}$$

$$\frac{q : P_n \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \qquad n \geq 1}{q ; a : P_n}\ \textsc{p-sib}$$

2.1.2 Proof of Inference Rules

Definition 2.1.1 (Domination of evaluation plans). *An evaluation plan q is said to* dominate *another evaluation plans q' if it holds that for all XML documents D and nodes $n \in N_D$ it holds that $\llbracket q' \rrbracket_D(n) \subseteq \llbracket q \rrbracket_D(n)$.*

Definition 2.1.2 (Monotonicity of properties). *A property π is said to be monotonous if for for all evaluation plans $q, q' \in \mathcal{I}$ it holds that if $q : \pi$ and q dominates q' then it follows that $q' : \pi$*

Lemma 2.1.1 (Monotonicity of positive set properties). *All positive set properties are monotone.*

Proof. Since positive set properties forbid certain configurations in the result of an evaluation plan, it holds that if q has this property and q' always returns a subset of the result of q, then q' will also have this property. □

Lemma 2.1.2. *Given two evaluation plans q and q' it holds that if q dominates q' then $q; \uparrow$ dominates $q'; \uparrow$.*

Proof. Let D be an XML document and $n \in N_D$. If $n' \in \llbracket q'; \uparrow \rrbracket^*_D(n)$ then there is a child n'' of n' in $\llbracket q' \rrbracket^*_D(n)$, and since q dominates q' it follows that $n'' \in \llbracket q \rrbracket^*_D(n)$ and therefore $n' \in \llbracket q; \uparrow \rrbracket^*_D(n)$. □

Rules for *ord*

ORD-NO2D-STEP (2.1.1)

$$\frac{q : no2d, nodup1 \quad a \in A}{q; a : ord}$$

Proof. If *nod2d* and *nodup* hold then the result is a single node and, by definition, axes applied to a single node result in an ordered sequence of nodes. ☐

ORD-NORC-PRN (2.1.1)

$$\frac{q : norc, ord}{q; \uparrow : ord}$$

Proof. Let a and b be two input nodes of D with $a \prec b$ and let c and d be their respective parent nodes in the result of the step expression. Suppose now that $d \prec c$ This means that either d is an ancestor of c or d is a preceding node of c.

- **d is a preceding node of c** – If this were true, then all children of d would be preceding nodes of c. This however conflicts with the fact that $a \prec b$;
- **d is an ancestor node of c** – Since $a \prec b$, this implies that there is an ancestor of c that has a right sibling, namely b. This conflict with the definition of the *norc* property.

We conclude that any two parents of two input nodes occur in order in the result. ☐

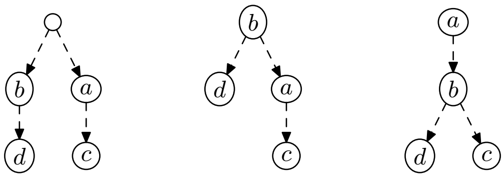
ORD-UNREL-NODUP-DOWN (2.1.1)

$$\frac{\begin{array}{l} q : unrel, ord \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q; a : ord}$$

Proof. Let a and b be two input nodes of D. Then a and b are two different nodes since we have no duplicates in the input sequence. Hence we can assume without loss of generality that $a \prec b$. Let c and d be their respective child nodes in the result of the step expression. Suppose now that $d \prec c$ This means that either d is an ancestor of c or d is a preceding node of c.

d **is an ancestor node of** c – This implies that b is an ancestor of a or vice versa which is in conflict with the *unrel* property.

d **is a preceding node of** c – There are three possibilities in this case:



(i) (ii) (iii)

- **(i)** b **is a preceding node of** a – This conflicts with $a \prec b$;
- **(ii)** b **is a ancestor node of** a – This conflicts with the *unrel* property;
- **(iii)** b **is a descendant node of** a **and an ancestor node of** c – This conflicts with the *unrel* property.

We conclude that $c \prec d$ for every two nodes c and d in the result of the step expression. ☐

Rules for *nodup*

NODUP-NO2D-STEP (2.1.1)

$$\frac{\begin{array}{l} q : no2d, nodup \\ a \in A \end{array}}{q; a : nodup}$$

Proof. If *no*2*d* and *nodup* hold for q then the result of q contains at most one node, and it follows from the definition of the evaluation plan of an axis that the axis applied to a single node contains no duplicates. ☐

NODUP-CHL (2.1.1)

$$\frac{q : nodup}{q; \downarrow : nodup} \text{ NODUP-CHL}$$

Proof. Since the nodes are part of a tree, no two different nodes will have a common child. Hence if a certain node would occur twice in the result after following the child axis, then this would imply that it's parent occured twice in the input list. This cannot be, because the *nodup* property holds. ☐

NODUP-LIN-PRNSIB (2.1.1)

$$\frac{q : \mathit{lin}, \mathit{nodup}}{q; \uparrow : \mathit{nodup}}$$

Proof. Only duplicate nodes or pairs of siblings will produce duplicates in the list of parents or siblings. Since both the *nodup* and the *lin* property hold, no duplicates will occur in the result after following the parent, preceding sibling or following sibling axis. □

NODUP-UNREL-DOWN (2.1.1)

$$\frac{\begin{array}{l} q : \mathit{unrel}, \mathit{nodup} \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q; a : \mathit{nodup}}$$

Proof. By definition of a tree, two unrelated nodes never have common descendants. Therefore, unrelated nodes will never generate duplicates when one of the axes $\downarrow$, $\downarrow^+$ or $\downarrow^*$ are followed. □

Rules for *no2d*

NO2D-UP (2.1.1)

$$\frac{q : \mathit{no2d}_n \qquad n \geq 0}{q : \mathit{no2d}_{n+1}}$$

Proof. If a sequence has the same n^{th} ancestor, then also the $(n+1)^{th}$ ancestor is the same, since the n^{th} ancestor has at most one parent. □

Rules for *lin*

LIN-NO2D (2.1.1)

$$\frac{q : \mathit{no2d}_n \qquad n \geq 0}{q : \mathit{lin}}$$

Proof. If a set contains no two distinct nodes then all the nodes in the sequence are the same, and hence linear. Suppose that $\mathit{no2d}_n$ with $n > 0$ holds. Then for the n^{th} ancestors the *no2d* holds and hence *lin* holds for the n^{th} ancestors of the sequence. This is equal to saying that lin_n holds for the sequence. □

LIN-UP (2.1.1)

$$\frac{q : \mathit{lin}_n \qquad n \geq 0}{q : \mathit{lin}_{n+1}}$$

Proof. If a sequence of nodes of a tree are linear then their parents are also linear. Hence if the n^{th} ancestors of the nodes of node sequence are linear then also their $(n+1)^{th}$ ancestors are linear. □

LIN-ANC (2.1.1)

$$\frac{q : lin_n \qquad n \geq 1}{q; \uparrow^{+} : lin_{n-1}}$$

Proof. Suppose that the lin_{n-1} property does not hold after following the ancestor axis. Then there are two nodes n_1 and n_2, which are respectively the m_1^{th} and m_2^{th} ancestor of nodes n_3 and n_4 of the input sequence $(m_1, m_2 > 1)$ and for which their $(n-1)^{th}$ ancestors (respectively n_5 and n_6) are not linear. Since the lin_n property holds in the input sequence, it follows from LIN-UP (2.1.1) that the all ancestors that are n or more levels upwards from n_3 and n_4 are linear. Since the ancestor relation is transitive we know that n_5 and n_6 are respectively the $(m_1+(n-1))^{th}$ and $(m_2+(n-1))^{th}$ ancestors of n_3 and n_4. Hence we have two ancestors that are n or more levels upwards from the input nodes and that are not linear. This is a contradiction with the assumption that the lin_n property holds in the input sequence. ☐

LIN-AOS (2.1.1)

$$\frac{q : lin_n \qquad n \geq 0}{q; \uparrow^{*} : lin_n}$$

Proof. The proof of this inference rule is analogous to that of LIN-ANC (2.1.1). ☐

Rules for *nolc*

NOLC-LIN (2.1.1)

$$\frac{q : lin}{q : nolc}$$

Proof. If all nodes in a sequence are linear then they do not have siblings in the sequence and therefore the *nolc* property holds. ☐

NOLC-FLS (2.1.1)

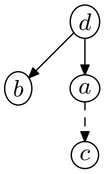
$$\frac{q : nolc}{q; \overset{.}{\twoheadrightarrow} : nolc}$$

Proof. Suppose that after following the following-sibling axis the *nolc* property does not hold anymore. Then there are two nodes n_2, n_3 in the result sequence such that n_2 has a sibling n_4 that is an ancestor of n_3 and n_2 is a left sibling of n_4. Since n_2 and n_3 are in the result sequence after selecting the following siblings, we know that there have to be left siblings of n_2 and n_3 (respectively n_5 and n_6) in the input sequence. But then n_5 is also a left sibling of n_4 and n_4 is an ancestor of n_6. Hence the *nolc* property does not hold in the input sequence, which contradicts the initial assumption. ☐

NOLC-LIN-AOS (2.1.1)

$$\frac{\begin{array}{l} q : nolc_n, lin_{n+1} \\ n \geq 0 \end{array}}{q; \uparrow^* : nolc_n}$$

Proof. Suppose $n = 0$. Then $nolc$ and lin_1 hold in the input sequence. Now suppose that $nolc$ does not hold after following the $\uparrow^*$ axis. Hence there are two nodes b and c in the result which are positioned as follows:



Since b is not on the line, it has to be in the input sequence (this follows from lin_1), and since c is in the output sequence, c or a descendant of c has to be in the input sequence. Hence $nolc$ does not hold in the input sequence. Suppose $n > 0$. Then for the n^{th} ancestors $nolc$ and lin_1 holds and hence $nolc$ holds for the n^{th} ancestors after following the $\uparrow^*$ axis and therefore $nolc_n$ holds in the output sequence. ☐

NOLC-LIN-ANC (2.1.1)

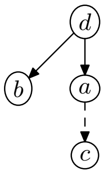
$$\frac{\begin{array}{l} q : nolc_n, lin_{n+1} \\ n \geq 1 \end{array}}{q; \uparrow^+ : nolc_{n-1}}$$

Proof. The result of $\uparrow^+$ is equal to the result of $\uparrow; \uparrow^*$. From rule P-PRN (2.1.1) follows that after following the $\uparrow$ axis, $nolc_{n-1}$ and lin_n holds. Hence it follows by rule NOLC-LIN-AOS (2.1.1) that by following the $\uparrow^*$ axis after the $\uparrow$ axis, the property $nolc_{n-1}$ holds. ☐

NOLC-UNREL1-SIB (2.1.1)

$$\frac{\begin{array}{l} q : unrel_1 \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : nolc}$$

Proof. Suppose that after following the $\dot{\twoheadleftarrow}$ or $\dot{\twoheadrightarrow}$ axis the $nolc$ property doesn't hold. Then we have two nodes b and c which are positioned according to following figure:

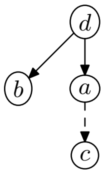


It is clear that $unrel_1$ does not hold for the result, since the parents of b and c are related. But since the set of parents of the output sequence is a subset of those of the input sequence, $unrel_1$ does not hold in the input sequence and hence we get a contradiction. ☐

NOLC-UNREL-CHL (2.1.1)

$$\frac{q : unrel}{q; \downarrow : nolc}$$

Proof. Suppose that after following the ↓ axis the *nolc* property doesn't hold. Then we have two nodes b and c which are positioned according to following figure:



But then the parents of b and c are in the input sequence and hence *unrel* does not hold in the input sequence. ☐

NOLC-NO2D (2.1.1)

$$\frac{q : no2d_n \qquad n \geq 0}{q : nolc_{\geq 0}}$$

Proof. Since $no2d_n$ holds for some $n \geq 0$, we know that all nodes in the sequence are the n^{th} descendants of one node and hence are on the same level. Therefore it is impossible to have two nodes of a different level in the sequence and hence there are no two nodes b and c such that a sibling of b is an ancestor of c. ☐

Rules for *norc*

NORC-LIN (2.1.1)

$$\frac{q : lin}{q : norc}$$

Proof. If all nodes in a sequence are linear then they do not have siblings in the sequence and therefore the *norc* property holds. ☐

NORC-PRS (2.1.1)

$$\frac{q : norc}{q; \dot{\twoheadleftarrow} : norc}$$

Proof. Suppose that after following the preceding-sibling axis the *norc* property does not hold anymore. Then there are two nodes n_2, n_3 in the result sequence such that n_2 has a sibling n_4 that is an ancestor of n_3 and n_2 is a right sibling of n_4. Since n_2 and n_3 are in the result sequence after selecting the preceding siblings, we know that there have to be right siblings of n_2 and n_3 (respectively n_5 and n_6) in the input sequence. But then n_5 is also a right sibling of n_4 and n_4 is an ancestor of n_6. Hence the *norc* property does not hold in the input sequence, which contradicts the initial assumption. ☐

NORC-LIN-AOS (2.1.1)

$$\frac{\begin{array}{l} q : norc_n, lin_{n+1} \\ n \geq 1 \end{array}}{q; \uparrow^* : norc_n}$$

Proof. Analogous to the proof of NOLC-LIN-AOS (2.1.1). ☐

NORC-LIN-ANC (2.1.1)

$$\frac{q : norc_n, lin_{n+1} \quad n \geq 1}{q; \uparrow^+ : norc_{n-1}}$$

Proof. Analogous to the proof of NOLC-LIN-ANC (2.1.1). ☐

NORC-UNREL1-SIB (2.1.1)

$$\frac{q : unrel_1 \quad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : norc}$$

Proof. Analogous to the proof of NOLC-UNREL1-SIB (2.1.1). ☐

NORC-UNREL-CHL (2.1.1)

$$\frac{q : unrel}{q; \downarrow : norc}$$

Proof. Analogous to the proof of NOLC-UNREL-CHL (2.1.1). ☐

NORC-NO2D (2.1.1)

$$\frac{q : no2d_n \qquad n \geq 0}{q : norc_{\geq 0}}$$

Proof. Analogous to the proof of NOLC-NO2D (2.1.1) ☐

Rules for *unrel*

UNREL-NOLC-FLS (2.1.1)

$$\frac{q : nolc}{q; \dot{\twoheadrightarrow} : unrel}$$

Proof. Suppose that after following the following-sibling axis the *unrel* property does not hold. Then there are two nodes n_1 and n_2 in the output sequence such that n_1 is an ancestor of n_2. Hence both n_1 and n_2 have a left sibling which is in the input sequence, say n_3 and n_4. Hence n_3 has a sibling n_1 that is an ancestor of n_2 and hence n_4 (since siblings have the same ancestors). Furthermore n_3 is a left sibling of n_1. Therefore the *nolc* property does not hold in the input sequence which contradicts our initial assumption. ☐

UNREL-NORC-PRS (2.1.1)

$$\frac{q : norc}{q; \dot{\twoheadleftarrow} : unrel}$$

Proof. Suppose that after following the preceding-sibling axis the *unrel* property does not hold. Then there are two nodes n_1 and n_2 in the output sequence such that n_1 is an ancestor of n_2. Hence both n_1 and n_2 have a right sibling which is in the input sequence, say n_3 and n_4. Hence n_3 has a sibling n_1 that is an ancestor of n_2 and hence n_4 (since siblings have the same ancestors). Furthermore n_3 is a right sibling of n_1. Therefore the *norc* property does not hold in the input sequence which contradicts our initial assumption. ☐

UNREL-DOWN (2.1.1)

$$\frac{q : unrel_n \qquad n \geq 1}{q : unrel_{n-1}}$$

Proof. Suppose that the $unrel_n$ property holds and the $unrel_{n-1}$ property doesn't. Then there are two nodes n_1 and n_2 for which their $(n-1)^{th}$ ancestors are related and their n^{th} ancestors are not, which is a contradiction. ☐

UNREL-NO2D (2.1.1)

$$\frac{q : no2d_n \qquad n \geq 0}{q : unrel}$$

Proof. Suppose that the $no2d_n$ property holds and the *unrel* property doesn't. Then there are two nodes on the same level which have an ancestor-descendant relation, which is clearly a contradiction. ☐

Rules for Set Properties

P-CHL (2.1.1)

$$\frac{q : P_n \qquad n \geq 0}{q; \downarrow : P_{n+1}}$$

Proof. For every node n of a document D, it holds that $[\![q; \downarrow; \uparrow]\!]_D(n) \subseteq [\![q]\!]_D(n)$. Since the nodes in q are a subset of those in $q; \downarrow; \uparrow$ (up to some duplicates), we know (by Lemma 2.1.1) that if q has the P property, so does $q; \downarrow; \uparrow$. This implies (by Definition 1.2.10) that $q; \downarrow$ has the P_{n+1} property. ☐

P-PRN (2.1.1)

$$\frac{q : P_n \qquad n \geq 1}{q; \uparrow : P_{n-1}}$$

Proof. This follows from Definition 1.2.10. ☐

P-SIB (2.1.1)

$$\frac{\begin{array}{l} q : P_n \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow} \qquad n \geq 1\} \end{array}}{q; a : P_n}$$

Proof. If every node of the input sequence has a left of right sibling then the n^{th} ancestors (with $n \geq 1$) remain the same after following respectively the preceding-sibling and following-sibling axis. Else the n^{th} ancestors of the output sequence is a subset of the n^{th} ancestors from the input sequence, for which the P property holds. Hence it follows by Lemma 2.1.1 that also the P property holds for the n^{th} ancestors of the output sequence and hence P_n holds for the output sequence. ☐

2.2 Completeness Rules

In this section and the next one, we show that the set of rules we presented in Section 2.1 is complete. We do this by proving that for each evaluation plan q for which we cannot derive the *ord* and *nodup* properties, there exists at least one document D such that for some node n of D it holds that $[\![q]\!]_D(n)$ is out of document order or contains duplicates; i.e. that the evaluation plan has the $\neg ord$ or $\neg nodup$ property.

2.2.1 Inference Rules

In order to be able to derive possible unorderedness or duplicate generation for an evaluation plan we will need an additional set of inference rules that enable us to derive the required negative properties. Again, we first present the rules and prove their soundness afterwards.

Rules for $\neg \mathit{ord}$

$$\frac{\begin{array}{l} q : \neg \mathit{no2d} \\ a \in \{\twoheadleftarrow, \twoheadrightarrow, \uparrow^*, \uparrow^+\} \end{array}}{q; a : \neg \mathit{ord}} \ \textsc{not-ord-no2d-fpa}$$

$$\frac{\begin{array}{l} q : \neg \mathit{unrel} \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q; a : \neg \mathit{ord}} \ \textsc{not-ord-unrel-down}$$

$$\frac{q : \neg \mathit{norc}, \mathit{ord}}{q; \uparrow : \neg \mathit{ord}} \ \textsc{not-ord-norc-prn}$$

$$\frac{q : \mathit{nsib} \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : \neg \mathit{ord}} \ \textsc{not-ord-nsib-sib}$$

$$\frac{q : \neg \mathit{unrel}, \mathit{ord}}{q; \dot{\twoheadrightarrow} : \neg \mathit{ord}} \ \textsc{not-ord-unrel-fls}$$

Rules for $\neg \mathit{nodup}$

$$\frac{\begin{array}{l} q : \neg \mathit{no2d} \\ a \in \{\twoheadleftarrow, \twoheadrightarrow, \uparrow^*, \uparrow^+\} \end{array}}{q; a : \neg \mathit{nodup}} \ \textsc{not-nodup-no2d-fpa}$$

$$\frac{\begin{array}{l} q : \neg \mathit{unrel} \\ a \in \{\downarrow^*, \downarrow^+\} \end{array}}{q; a : \neg \mathit{nodup}} \ \textsc{not-nodup-unrel-dsc}$$

$$\frac{\begin{array}{l} q : \mathit{nsib} \\ a \in \{\uparrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : \neg \mathit{nodup}} \ \textsc{not-nodup-nsib-prnsib}$$

Rules for ntree

$$\frac{a \in \{\downarrow^*, \downarrow^+, \twoheadleftarrow, \twoheadrightarrow\}}{q; a : \mathit{ntree}} \ \textsc{ntree-sink}$$

$$\frac{q : \mathit{ntree} \qquad a \in A}{q; a : \mathit{ntree}} \ \textsc{ntree-step}$$

Rules for $\neg \mathit{unrel}$

$$\frac{a \in \{\uparrow^*, \uparrow^+\}}{q; a : \neg \mathit{unrel}} \ \textsc{not-unrel-anc}$$

$$\frac{q : \neg \mathit{unrel} \qquad a \in \{\downarrow, \uparrow\}}{q; a : \neg \mathit{unrel}} \ \textsc{not-unrel-prnchl}$$

$$\frac{q : \neg \mathit{norc}}{q; \dot{\twoheadleftarrow} : \neg \mathit{unrel}} \ \textsc{not-unrel-norc-prs}$$

$$\frac{q : \neg \mathit{nolc}}{q; \dot{\twoheadrightarrow} : \neg \mathit{unrel}} \ \textsc{not-unrel-nolc-fls}$$

$$\frac{q : \mathit{ntree}}{q : \neg \mathit{unrel}} \ \textsc{not-unrel-ntree}$$

Rules for $\neg \mathit{no2d}$

$$\frac{a \in \{\uparrow^*, \uparrow^+\}}{q; a : \neg \mathit{no2d}_{\geq 0}} \ \textsc{not-no2d-anc}$$

$$\frac{q : \neg \mathit{no2d}_{\geq 0} \qquad a \in A}{q; a : \neg \mathit{no2d}_{\geq 0}} \ \textsc{not-no2d-step}$$

$$\frac{q : \mathit{nsib}}{q : \neg \mathit{no2d}} \ \textsc{not-no2d-nsib}$$

Rules for nsib

$$\frac{a \in \{\downarrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : \mathit{nsib}} \ \textsc{nsib-chlsib}$$

$$\frac{q : \mathit{nhat}_n \qquad n \geq 0}{q : \mathit{nsib}_n} \ \textsc{nsib-nhat}$$

Rules for $\neg nolc$

$$\frac{q : nhat_n \qquad n \geq 0}{q : \neg nolc_n}\;\textsc{not-nolc-nhat} \qquad\qquad \frac{q : \neg unrel}{q ; \dot{\twoheadleftarrow} : \neg nolc}\;\textsc{not-nolc-unrel-prs}$$

$$\frac{q : \neg nolc}{q ; \dot{\twoheadleftarrow} : \neg nolc}\;\textsc{not-nolc-prs}$$

Rules for $\neg norc$

$$\frac{q : nhat_n \qquad n \geq 0}{q : \neg norc_n}\;\textsc{not-norc-nhat} \qquad\qquad \frac{q : \neg unrel}{q ; \dot{\twoheadrightarrow} : \neg norc}\;\textsc{not-norc-unrel-fls}$$

$$\frac{q : \neg norc}{q ; \dot{\twoheadrightarrow} : \neg norc}\;\textsc{not-norc-fls}$$

Rules for $nhat$

$$\frac{q : \neg unrel}{q ; \downarrow : nhat}\;\textsc{nhat-unrel-chl} \qquad\qquad \frac{q : nhat \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q ; a : nhat}\;\textsc{nhat-sib}$$

$$\frac{q : \neg nolc}{q ; \dot{\twoheadrightarrow} : nhat}\;\textsc{nhat-nolc-fls} \qquad\qquad \frac{q : \neg norc}{q ; \dot{\twoheadleftarrow} : nhat}\;\textsc{nhat-norc-prs}$$

$$\frac{q : nsib_n \qquad n \geq 1}{q ; \uparrow^* : nhat_{n-1}}\;\textsc{nhat-nsib-aos} \qquad\qquad \frac{q : nsib_n \qquad n \geq 2}{q ; \uparrow^+ : nhat_{n-2}}\;\textsc{nhat-nsib-anc}$$

$$\frac{q : ntree}{q : nhat}\;\textsc{nhat-ntree}$$

Rules for Negative Set Properties

In order to avoid having separate rules for all negative set properties with an index, we provide some general properties that apply to all negative set properties. Hence if $NP \in \{\neg no2d, \neg unrel, nsib,$line$break \neg norc, nhat,$ then following rules hold:

$$\frac{q : NP_n \qquad n \geq 0}{q ; \downarrow : NP_{n+1}}\;\textsc{np-chl} \qquad\qquad \frac{q : NP_n \qquad a \in \{\uparrow, \uparrow^+\} \qquad n \geq 1}{q ; a : NP_{n-1}}\;\textsc{np-prnanc}$$

$$\frac{q : NP_n \qquad n \geq 0}{q ; \uparrow^* : NP_n}\;\textsc{np-aos} \qquad\qquad \frac{q : NP_n \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \qquad n \geq 1}{q ; a : NP_n}\;\textsc{np-sib}$$

2.2.2 Proof of Inference Rules

Definition 2.2.1 (Anti-monotonicity of properties). *A property π is said to be* anti-monotonous *if for for all evaluation plans $q, q' \in \mathcal{I}$ it holds that if $q : \pi$ and q' dominates q then it follows that $q' : \pi$*

Lemma 2.2.1 (Anti-monotonicity of negative set properties). *All negative set properties are anti-monotone.*

Proof. Since negative set properties require that certain configurations sometimes appear in the result of an evaluation plan, it holds that if q has such property and q' always returns a superset of the result of q, then q' will also have this property. $\square$

Rules for $\neg ord$

NOT-ORD-NO2D-FPA
(2.2.1)

$$\frac{\begin{array}{l} q : \neg no2d \\ a \in \{\twoheadleftarrow, \twoheadrightarrow, \uparrow^*, \uparrow^+\} \end{array}}{q; a : \neg ord}$$

Proof. Suppose that $\neg no2d$ holds for an evaluation plan. Then there is a document such that result of the evaluation plan is a sequence that contains two distinct nodes a and b. Since a document is a tree, each pair of distinct nodes have a sequence $(c, d, \ldots)$ of ancestors in common and hence after following the $\uparrow^*$ or $\uparrow^+$ axis, this sequence occurs at least twice, without being merged. Suppose (without loss of generality) that $c \prec d$. Since the sequence occurs at least twice in the output, the first d comes before the second c in the result and hence the result is not in document order. Furthermore there is a document in which d has two left siblings (e and f). These left siblings are preceding nodes of a and b and hence there is an a that comes before a b which in turn comes before another occurence of a in the result sequence of $\twoheadleftarrow$. Hence the sequence after following the $\twoheadleftarrow$ axis is not in document order. Analogously, following the $\twoheadrightarrow$ axis results in a sequence that is not in document order if the input contains two or more distinct nodes. ☐

NOT-ORD-UNREL-DOWN
(2.2.1)

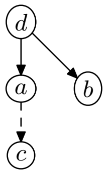
$$\frac{\begin{array}{l} q : \neg unrel \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q; a : \neg ord}$$

Proof. Since the $\neg unrel$ property holds for the input, there is a document such that the input sequence contains two related nodes a and b (let a be an ancestor of b). Hence there is a node c which is ancestor-or-self of b and child of a and the document can also contain a left and a right sibling of c, resp. d and e. Hence $a \prec d \prec c \prec b \prec e$. Since the e is a follower of every child f of b, we know for every f (child of b) that $b \prec f \prec e$. If a comes before b in the input sequence then e (which is a child of a) comes before f in the output sequence and hence the output sequence is not in document order. Else if b comes before a in the input sequence then f comes before d (which is a child of a) and also in this case the output sequence is not in document order. Hence $\neg ord$ holds for the output of $\downarrow$, $\downarrow^*$, and $\downarrow^+$, if $\neg unrel$ holds for the input. ☐

NOT-ORD-NORC-PRN
(2.2.1)

$$\frac{q : \neg norc, ord}{q; \uparrow : \neg ord}$$

Proof. The $\neg norc$ property implies that there is a document such that after following q the input sequence contains two nodes b and c, which are structured as follows:



Since $c \prec b$ and the ord property holds for the input, we know that c comes before b in the input sequence. Suppose the parent of c is e. Then e comes before d (which is the parent of b) in the output sequence, but from the fact that e is a descendant of d follows that $d \prec e$. Hence the output sequence is not in document order. ☐

NOT-ORD-NSIB-SIB (2.2.1)

$$\frac{q : \mathit{nsib} \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : \neg \mathit{ord}}$$

Proof. Since q has the *nsib* property, there is a document for which the evaluation of the evaluation plan q results in a sequence containing a node a with n siblings (for any n). Now suppose that a has three right siblings, b, c and d (in this order), in the input sequence. Then after following the $\dot{\twoheadleftarrow}$ axis, the output sequence will contain the nodes a, a, b, a, b, c in that order and hence the output sequence is not in document order (since the last a comes before the first b). Analogously the result is not in document order after following the $\dot{\twoheadrightarrow}$ axis. ☐

NOT-ORD-UNREL-FLS (2.2.1)

$$\frac{q : \neg \mathit{unrel}, \mathit{ord}}{q; \dot{\twoheadrightarrow} : \neg \mathit{ord}}$$

Proof. From the $\neg$*unrel* property we know that there is a document such that the evaluation of q contains two nodes a and b where a is an ancestor of b. Hence a comes before b in document order, and since *ord* holds also in the input sequence. Therefore after following the $\dot{\twoheadrightarrow}$ axis the right siblings of a come before the right siblings of b in the output sequence. But since the siblings of b are also descendants of a, those nodes come in document order before the the next sibling of a. Hence the result sequence is not in document order. ☐

Rules for $\neg$*nodup*

NOT-NODUP-NO2D-FPA (2.2.1)

$$\frac{\begin{array}{l} q : \neg \mathit{no2d} \\ a \in \{\twoheadleftarrow, \twoheadrightarrow, \uparrow^*, \uparrow^+\} \end{array}}{q; a : \neg \mathit{nodup}}$$

Proof. Suppose that $\neg$*no*2*d* holds for an evaluation plan. Then there is a document such that result of the evaluation plan is a sequence that contains two distinct nodes a and b. Since a document is a tree, each pair of distinct nodes have ancestors in common and hence after following the $\uparrow^*$ or $\uparrow^+$ axis the $\neg$*nodup* property holds. Suppose c is a common ancestor of a and b. Then there is a document in which c has a left sibling. This left sibling is a preceding node of a and b and hence $\neg$*nodup* holds after following the $\twoheadleftarrow$ axis. Analogously, there is a document with a right sibling of c which is obviously a following node of both a and b. ☐

NOT-NODUP-UNREL-DSC (2.2.1)

$$\frac{\begin{array}{l} q : \neg \mathit{unrel} \\ a \in \{\downarrow^*, \downarrow^+\} \end{array}}{q; a : \neg \mathit{nodup}}$$

Proof. Since in the input the $\neg$*unrel* property holds, we know that it is possible to have an input sequence such that there are two different nodes a and b that are related. Suppose (without loss of generality) that a is an ancestor of b. Then all descendants of b are also descendants of a and hence these descendants appear twice in the output sequence. ☐

NOT-NODUP-NSIB-PRNSIB (2.2.1)

$$\frac{\begin{array}{l} q : nsib \\ a \in \{\uparrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : \neg nodup}$$

Proof. Since q has the *nsib* property, there is a document for which the evaluation of the evaluation plan q results in a sequence containing a node a with n siblings (for any n). Now suppose that a has two right siblings, b and c (in that order), in the input sequence. Then after following the $\dot{\twoheadleftarrow}$ axis, a will occur at least twice in the result sequence and after following the $\dot{\twoheadrightarrow}$ axis, c will occur at least twice in the result sequence. Furthermore, since siblings have the same parent, it is obvious that duplicates occur after following the $\uparrow$ axis. Hence, if the *nsib* property holds for the input evaluation plan, then $\neg$*nodup* holds after following the $\uparrow$, $\dot{\twoheadleftarrow}$ or $\dot{\twoheadrightarrow}$ axis. ☐

Rules for *ntree*

NTREE-SINK (2.2.1)

$$\frac{a \in \{\downarrow^*, \downarrow^+, \twoheadleftarrow, \twoheadrightarrow\}}{q; a : ntree}$$

Proof. For each evaluation plan q there is a document with a node a in the result of q that has a left and a right sibling (respectively b and c) in the document such that a, b, and c have a tree of size n beneath (which both do not have to be in the result of q). Hence after following the $\downarrow^*$ or $\downarrow^+$ axis the *ntree* property holds (since there is a tree of size n beneath a) and also after following the $\twoheadleftarrow$ or $\twoheadrightarrow$ axis the *ntree* property holds (since there is a tree of size n beneath respectively b and c). ☐

NTREE-STEP (2.2.1)

$$\frac{q : ntree \qquad a \in A}{q; a : ntree}$$

Proof. For the $\downarrow^*$, $\downarrow^+$, $\twoheadleftarrow$ and $\twoheadrightarrow$ axis it clearly holds by Rule NTREE-SINK that the *ntree* property holds afterwards. Since the *ntree* property holds, there is a document such that the evaluation of q contains a tree of size n. After following the $\downarrow, \uparrow, \uparrow^*, \uparrow^+, \dot{\twoheadleftarrow}$ and $\dot{\twoheadrightarrow}$ axis we then get a tree of size $n - 1$ and hence the *ntree* property holds after following these axis, since we assumed that for any n there was a document such that there is a tree of size n in the result of q (and hence also for $n + 1$). ☐

Rules for $\neg$*unrel*

NOT-UNREL-ANC (2.2.1)

$$\frac{a \in \{\uparrow^*, \uparrow^+\}}{q; a : \neg unrel}$$

Proof. All ancestors of a node are related and hence $\neg$*unrel* holds after following the $\uparrow^*$ of $\uparrow^+$ axis. ☐

NOT-UNREL-PRNCHL (2.2.1)

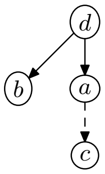
$$\frac{q : \neg \mathit{unrel} \qquad a \in \{\downarrow, \uparrow\}}{q; a : \neg \mathit{unrel}}$$

Proof. Suppose that $\neg \mathit{unrel}$ holds in the input. Then there is an XML document such that there are two related and distinct nodes a and b in the input sequence. Suppose (without loss of generality) that a is an ancestor of b. If we follow the $\uparrow$ axis then it is obvious that the parent of b is a or an ancestor of a and hence the parent of a is an ancestor of the parent of b. If we follow the $\downarrow$ axis then the there is a child of a that is b or an ancestor of b and hence an ancestor of every child of b. In both cases we have at least two related nodes after following the axis and hence $\neg \mathit{unrel}$ holds. ☐

NOT-UNREL-NOLC-FLS (2.2.1)

$$\frac{q : \neg \mathit{nolc}}{q; \dot{\twoheadrightarrow} : \neg \mathit{unrel}}$$

Proof. Suppose $\neg \mathit{nolc}$ holds in the input. Then there is an XML document such that the input sequence contains at least two nodes b and c which are positioned as follows:



After following the $\dot{\twoheadrightarrow}$ axis, we get the node a in the result sequence and all right siblings of c (possibly among other nodes). But since siblings have the same ancestors, we know that all right siblings of c are descendants of a and hence there are two nodes a and e (where e is a right sibling of c) which are related. Hence $\neg \mathit{unrel}$ holds in the result sequence. ☐

NOT-UNREL-NORC-PRS (2.2.1)

$$\frac{q : \neg \mathit{norc}}{q; \dot{\twoheadleftarrow} : \neg \mathit{unrel}}$$

Proof. Analogous to the proof of NOT-UNREL-NOLC-FLS (2.2.1). ☐

NOT-UNREL-NTREE (2.2.1)

$$\frac{q : \mathit{ntree}}{q : \neg \mathit{unrel}}$$

Proof. If the *ntree* property holds for an evaluation plan then for any n there is a document such that after applying the path expression the result contains a tree of size n. It then follows from the definition of a tree that there are related nodes (i.e., $\neg \mathit{unrel}$ holds), since otherwise we have n roots and hence a forest. ☐

Rules for $\neg \mathit{no2d}$

NOT-NO2D-ANC (2.2.1)

$$\frac{a \in \{\uparrow^*, \uparrow^+\}}{q; a : \neg no2d_{\geq 0}}$$

Proof. For each q there is a document such that a node a is in the result of q and a has parent b and grandparent c (not necessarily in the result of q). Then b and c are both in the result of $\uparrow^*$ and $\uparrow^+$ for this document. Since c is the parent of b, the n^{th} ancestor of b is the $(n-1)^{th}$ ancestor of c. Hence there is no n such that the n^{th} ancestors of b and c are equal, and hence $\neg no2d_n$ holds for all n. ☐

NOT-NO2D-STEP (2.2.1)

$$\frac{q : \neg no2d_{\geq 0} \qquad a \in A}{q; a : \neg no2d_{\geq 0}}$$

Proof. Since $\neg no2d_{\geq 0}$ holds for q, there is a document D such that the evaluation of q contains two nodes a and b with different n^{th} ancestors for every n. If $n > 0$ then clearly the n^{th} ancestors of a and b, which were supposed to be different, can be both in the result. Else if $n = 0$ then it is obvious that, since a and b can have in number of siblings in D, $\neg no2d$ holds. Hence $\neg no2d_{\geq 0}$ holds for $q; \dot{\twoheadrightarrow}$ and $q; \dot{\twoheadleftarrow}$. Furthermore, since for each $n \geq 0$ the n^{th} ancestors of a and b are not equal, also the children and the parents of a and b have different n^{th} ancestors. Hence $\neg no2d_{\geq 0}$ holds for $q; \downarrow$ and $q; \uparrow$. Finally, it's also easy to see that after following the $\uparrow^*, \uparrow^+, \downarrow^*, \downarrow^+, \twoheadleftarrow$ and $\twoheadrightarrow$ axes it is possible to have nodes from a different level in the result, which implies that $\neg no2d_{\geq 0}$ holds after following these axes. ☐

NOT-NO2D-NSIB (2.2.1)

$$\frac{q : nsib}{q : \neg no2d}$$

Proof. Since the *nsib* property holds for q, there is for each n a document such that there is a node a which has at least n distinct siblings in the result. Hence each such document (for $n \geq 2$) has two (or more) distinct nodes, which implies that $\neg no2d$ holds for q. ☐

Rules for *nsib*

NSIB-CHLSIB (2.2.1)

$$\frac{a \in \{\downarrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : nsib}$$

Proof. For every node a there exists a document such that a has n children. Hence *nsib* holds after following the $\downarrow$ axis. Analogously, for every node a there exists a document such that a has n left (right) siblings and hence *nsib* holds after following the $\dot{\twoheadleftarrow}$ ($\dot{\twoheadrightarrow}$) aixs. ☐

NSIB-NHAT (2.2.1)

$$\frac{q : nhat_n \qquad n \geq 0}{q : nsib_n}$$

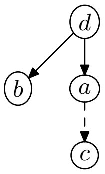
Proof. The *nhat* property implies that there is a document such that the evaluation of q results in a sequence containing a node a for which an ancestor b has at least n distinct left siblings and n distinct right siblings in the result sequence (for every n). Hence b has at least n distinct siblings in the result sequence (for every n), which is the definition of the *nsib* property. Now suppose $nhat_n$ for $n > 0$ holds for q. Then for the n^{th} ancestors the *nhat* and hence the *nsib* property holds. This implies that the $nsib_n$ property holds for q. ☐

Rules for $\neg nolc$

NOT-NOLC-NHAT (2.2.1)

$$\frac{q : nhat_n \qquad n \geq 0}{q : \neg nolc_n}$$

Proof. If *nhat* holds for an evaluation plan then for any n there is a document for which the result of the evaluation plan is containing a node c with an ancestor a such that a has at least n distinct left siblings and n distinct right siblings. Hence $\neg nolc$ holds, since there is a document for which the result of the evaluation plan contains following structure:



☐

NOT-NOLC-UNREL-PRS (2.2.1)

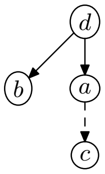
$$\frac{q : \neg unrel}{q; \dot{\twoheadleftarrow} : \neg nolc}$$

Proof. Suppose $\neg unrel$ holds in the input. Then there is a document such that there are two related nodes a and b, a descendant of a, in the input sequence. Suppose c and d are left siblings of respectively a and b. Then clearly c and d are in the output sequence and d has an ancestor a for which there is a left sibling (i.e. c) in the output sequence. Hence $\neg nolc$ holds in the result sequence. ☐

NOT-NOLC-PRS (2.2.1)

$$\frac{q : \neg nolc}{q; \dot{\twoheadleftarrow} : \neg nolc}$$

Proof. Suppose $\neg nolc$ holds in the input. Then there is a document such that there are two nodes b and c in the input sequence which are positioned as follows:



Now suppose the e is a left sibling of b and f is a left sibling of c. Then e and f are in the result and hence $\neg nolc$ holds in the output sequence (since a is an ancestor of f and e is a left sibling of a). ☐

Rules for $\neg norc$

NOT-NORC-NHAT (2.2.1)

$$\frac{q : nhat_n \qquad n \geq 0}{q : \neg norc_n}$$

Proof. Analogous to the proof of NOT-NOLC-NHAT (2.2.1). ☐

NOT-NORC-UNREL-FLS (2.2.1)

$$\frac{q : \neg unrel}{q; \dot{\twoheadrightarrow} : \neg norc}$$

Proof. Analogous to the proof of NOT-NOLC-UNREL-PRS (2.2.1). ☐

NOT-NORC-FLS (2.2.1)

$$\frac{q : \neg norc}{q; \dot{\twoheadrightarrow} : \neg norc}$$

Proof. Analogous to the proof of NOT-NOLC-PRS (2.2.1). ☐

Rules for *nhat*

NHAT-UNREL-CHL (2.2.1)

$$\frac{q : \neg unrel}{q; \downarrow : nhat}$$

Proof. Suppose that for q the $\neg unrel$ property holds. Then there is a document such that q has two related nodes a and b in the result where a is an ancestor of b. Hence there is a child c of a that is b or an ancestor of b and therefore c is an ancestor of every child d of b. Since a can have any number of children (both on the left hand and the right hand side of c), c can have at least n distinct left siblings and n distinct right siblings. Since all siblings of c and all children d of b are in the result after following the $\downarrow$ axis, the *nhat* property holds. ☐

NHAT-SIB (2.2.1)

$$\frac{q : nhat \qquad a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q; a : nhat}$$

Proof. The fact that *nhat* holds for q implies that for each n there is a document such that after the evaluation of q the result sequence contains a node a, which has an ancestor b such that at least n left siblings and at least n right siblings of b are also in the result sequence. Suppose $n = m + 1$. Then after following the $\dot{\twoheadleftarrow}$ and $\dot{\twoheadrightarrow}$ axis it holds that b has as least $n - 1 = m$ left siblings and at least $n - 1 = m$ right siblings for any n and hence for any m. Since the siblings of a have b as an ancestor too, it follows that *nhat* still holds after following the $\dot{\twoheadleftarrow}$ or $\dot{\twoheadrightarrow}$ axis. ☐

NHAT-NOLC-FLS (2.2.1)

$$\frac{q : \neg nolc}{q; \dot{\twoheadrightarrow} : nhat}$$

Proof. Since $\neg nolc$ holds for q, we know that there is a document D such that the result of q contains two nodes a and b where a is a left sibling of an ancestor of b. Let c be the sibling of a that is an ancestor of b. Then there can be any number n of siblings between a and c. Furthermore c can have any number of right siblings in the document D. Hence after following the $\dot{\twoheadrightarrow}$ axis there are at least n left siblings of c and at least n right siblings of c in the result sequence. Since c is an ancestor of every right sibling of b (which are also in the result sequence of $q; \dot{\twoheadrightarrow}$), the *nhat* property holds for $q; \dot{\twoheadrightarrow}$. ☐

NHAT-NORC-PRS (2.2.1)

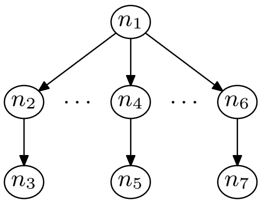
$$\frac{q : \neg norc}{q; \dot{\twoheadleftarrow} : nhat}$$

Proof. Analogous to the proof of NHAT-NOLC-FLS (2.2.1). ☐

NHAT-NSIB-AOS (2.2.1)

$$\frac{q : nsib_n \qquad n \geq 1}{q; \upharpoonright^* : nhat_{n-1}}$$

Proof. If q has the $nsib_1$ property, then there is a document such that after evaluating $q; \upharpoonright$ there exists a node with n siblings for any n. Therefore the children of this node and children of all the siblings of n have to be in the result after evaluating q. Hence in the result of q there are nodes n_3, n_5, n_7 which are positioned as follows:



We may assume that n_4 has at least n left siblings with children and at least n right siblings with children, since the $nsib_1$ property holds for q. After following the $\upharpoonright^*$ axis, the result contains n_5, n_4 and also at least n left siblings and n right siblings (since the result set of $\upharpoonright^*$ is a superset of the result set of $\upharpoonright$). Hence after following the $\upharpoonright^*$ axis, the *nhat* property holds.
Finally, if the $nsib_{n+1}$ property holds for q, then we know that for q followed by n parent steps the $nsib_1$ holds. Hence after following n times the $\upharpoonright$ axis and then the $\upharpoonright^*$ axis, we get *nhat*. But the result of following n times the $\upharpoonright$ axis and afterwards the $\upharpoonright^*$ axis is equal to the result of following the $\upharpoonright^*$ axis and afterwards n times the $\upharpoonright$ axis. Hence $nhat_n$ holds for q. ☐

NHAT-NSIB-ANC (2.2.1)

$$\frac{q : nsib_n \qquad n \geq 2}{q; \upharpoonright^+ : nhat_{n-2}}$$

Proof. We know that $\upharpoonright^+$ can be simulated by $\upharpoonright; \upharpoonright^*$. If the $nsib_n$ property holds for q then the $nsib_{n-1}$ holds for property holds for $q; \upharpoonright$ (this follows from Rule NP-PRNANC (2.2.1)). Rule NHAT-NSIB-AOS (2.2.1) then implies that $nhat_{n-2}$ holds for $q; \upharpoonright; \upharpoonright^*$ and hence for $q; \upharpoonright^+$. ☐

NHAT-NTREE (2.2.1)

$$\frac{q : ntree}{q : nhat}$$

Proof. Suppose that *ntree* holds for q, then there is a document such that we can have a tree of size $2n + 1$ in the result (for any n). Let a be the $(n + 1)^{th}$ child of the the root of this tree and b the $(n + 1)^{th}$ child of a. These nodes a and b exist in this document, since each node has at least $2n + 1$ children. We know that a is an ancestor of b and b has n left siblings and at least n right siblings. Hence (since n can be any positive integer) the *nhat* property holds for q. ☐

Rules for Negative Set Properties

NP-CHL (2.2.1)

$$\frac{q : NP_n \qquad n \geq 0}{q; \downarrow : NP_{n+1}}$$

Proof. Suppose D is a document which is used in showing that NP_n holds after the evaluation of q. Then we know that every extension of D can be used instead of D to show that NP_n holds after the evaluation of q. This is because every extension of D will result in a supersequence for the evaluation of q and it can be proven (analogously to the proof of Lemma 2.2.1) that this extended document is also an example. Therefore we will extend each document D in the proof of $q : NP_n$ to a new document D' by giving every node in D an extra child. By doing this, it follows that the sequence $\llbracket q; \downarrow; \uparrow \rrbracket_{D'}$ is a supersequence of $\llbracket q \rrbracket_{D'}$ (it may contain some extra duplicates). From $q : NP_n$ follows by Lemma 2.2.1 that $q; \downarrow; \uparrow : NP_n$ and hence $q; \downarrow : NP_{n+1}$ holds (Definition 1.2.10). $\square$

NP-PRNANC (2.2.1)

$$\frac{\begin{array}{l} q : NP_n \\ a \in \{\uparrow, \uparrow^+\} \qquad n \geq 1 \end{array}}{q; a : NP_{n-1}}$$

Proof. The fact that for $q; \uparrow$ the property NP_{n-1} holds, follows from Definition 1.2.10. Since the result of $q; \uparrow^+$ is a supersequence of $q; \uparrow$, the property NP_{n-1} also holds for $q; \uparrow^+$. $\square$

NP-AOS (2.2.1)

$$\frac{q : NP_n \qquad n \geq 0}{q; \uparrow^* : NP_n}$$

Proof. Since the result of q is a subsequence of the result of $q; \uparrow^*$, we know by Lemma 2.2.1 that NP_n holds for $q; \uparrow^*$ if NP_n holds for q. $\square$

NP-SIB (2.2.1)

$$\frac{\begin{array}{l} q : NP_n \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \qquad n \geq 1 \end{array}}{q; a : NP_n}$$

Proof. From the proof of Rule NP-CHL (2.2.1) we know that if D is used in showing that NP_n holds after the evaluation of q then it can be replaced in the proof of $q : NP_n$ by an extension of D. Therefore we will extend each document D in the proof of $q : NP_n$ to a new document D' by giving every node in D a left sibling. By doing this, it follows that the sequence $\llbracket q; \dot{\twoheadleftarrow}; \uparrow \rrbracket_{D'}$ is a supersequence of $\llbracket q; \uparrow \rrbracket_{D'}$ (it may contain some extra duplicates). From the fact that $q : NP_n$ follows that $q; \uparrow : NP_{n-1}$ (Definition 1.2.10) and hence (by Lemma 2.2.1) also $q; \dot{\twoheadleftarrow}; \uparrow : NP_{n-1}$. By Definition 1.2.10 then follows that $q; \dot{\twoheadleftarrow} : NP_n$. The proof for $\dot{\twoheadrightarrow}$ is analogous. $\square$

2.3 The A^{tidy} Automaton

From the inference rules, we can construct a deterministic automaton that decides whether *ord* and/or *nodup* hold after a certain tidy evaluation plan. The automaton serves two purposes: It is used to prove completeness, and it serves as the foundation for an evaluation plan.

Figure 2.1 and 2.2 present the (infinite) automaton, which we will call A^{tidy}. The start state is the state that contains *no*2*d*. The automaton should be thought of as continuing to the right indefinitely. Some states contain a name in brackets, such as l_0, l_1 l_1nlu l_1nru and s, which is used to duplicate them elsewhere in the diagram for readability. The state with name s is also referred to as the *sink state*.

The automaton should be used as follows. For a certain tidy evaluation plan we start in the initial state, which is labeled with *no*2*d*. Then we follow the transitions that correspond with the axes that are encountered in the evaluation plan from left to right. For example, for the tidy evaluation plan $\downarrow; \sigma; \delta; \downarrow; \sigma; \delta; \uparrow^+$; we start in the state labeled with $no2d$, then we follow the $\downarrow$ transition to the state with $no2d_1$. After that we follow the $\downarrow$ transition to the state with $no2d_2$ and finally we follow the $\uparrow^+$ transition to the state with lin_1, *norc* and *nolc*.

The properties that are contained in each state indicate which properties hold for the tidy evaluation plan that leads to that state. So for the example above, $q = \downarrow; \sigma; \delta; \downarrow; \sigma; \delta; \uparrow^+$, we can read from its final state that it holds that $q : lin_1$, $q : norc$, $q : nolc$, $q : \neg unrel$, $q : \neg ord_{\geq 0}$ and $q : nsib$. Also note that the automaton has an infinite number of states: From left to right, states are labeled with a π_i property where i strictly increases. Recall from Section 1.2.3 that if π_{i+1} holds for an evaluation plan q, then π_i holds for $q; \uparrow$. For example, applying $\downarrow$ in the *lin* state yields an expression in the lin_1 state, then applying $\uparrow$ in the lin_1 state returns to the *lin* state.

The type of the last edge (solid, dashed, dash-dotted or dotted) that is followed for the evaluation plan indicates whether the properties *ord* (dashed), *nodup* (dash-dotted), both (solid) or neither (dotted) hold. For example, we can read from the automaton that $\downarrow : ord, nodup$, $\downarrow; \sigma; \delta; \downarrow : ord, nodup$ but $\downarrow; \sigma; \delta; \downarrow; \sigma; \delta; \uparrow^+ : \neg ord, \neg nodup$. Consequently the type of the last edge indicates whether the tidy evaluation plan in question is correct up to ordering and correct up to duplicates.

The automaton confirms some intuitions about common path expressions. For instance, an arbitrary path of only child axes always yields results that are ordered and duplicate free. Recall the expression `$sur/procedure/desc::incision` from Section 1. A corresponding tidy evaluation plan is $\downarrow; \sigma; \delta; \downarrow^+$. Every transition for this evaluation plan is labeled by a solid arrow, meaning that none of the σ and δ operations are required. That is, the result $\downarrow; \sigma; \delta; \downarrow^+; \sigma; \delta$ is always equivalent to that of $\downarrow; \downarrow^+$ and has the same intermediate results after each axis step. The automaton also shows that the `//` idiom, which denotes the `desc-or-self::node()` axis, quickly leads to intermediate results that must be sorted. Consider the expression `$doc//a/b`, which abbreviates `$doc/desc-or-self::node()/child::a/child::b`. A corresponding evaluation plan is $\downarrow^*; \sigma; \delta; \downarrow; \sigma; \delta; \downarrow; \sigma; \delta$. From the automaton, we see the $\downarrow^*$ transition from the $no2d$ state to the sink state (labeled with (s)) yields a result that is *ord* and *nodup*, but all subsequent $\downarrow$ transitions in the sink state require sorting. So this evaluation plan can be rewritten as $\downarrow^*; \downarrow; \sigma; \downarrow; \sigma$, eliminating the first $\sigma; \delta$ and all subsequent δs. In Section 6, we discuss techniques for handling the `desc-or-self` axis.

2.3.1 Proving Soundness and Completeness

The automaton A^{tidy} allows us to easily prove the completeness of the set of inference rules for deriving the *ord* and *nodup* properties for tidy evaluation plans. The first step in this proof consists of showing that the nodes in the diagram of A^{tidy} are labeled with the "correct" properties and the edges in the diagram have the right type.

Theorem 2.3.1. *If for a tidy evaluation plan q the corresponding path in automaton A^{tidy} ends in a state that contains the property π then $q : \pi$.*

Proof. For each transition from state s_1 to state s_2, labeled with axis a, it can be shown that if the properties in s_1 hold for an evaluation plan q then the properties in s_2 can be derived with the reasoning rules presented in Section 2.1 and Section 2.2 for $q; a$, as is shown in Appendix A. Since all these properties are set properties it follows that if they hold for an evaluation plan q then they also hold for $q; \sigma; \delta$. It follows that if the properties in s_1 hold for a tidy evaluation plan q then the properties in s_2 hold for the tidy evaluation plan $q; \sigma; \delta; a$. It then follows with induction upon the number of axis in the tidy evaluation plan q that all the properties in the final state for q hold for q. $\square$

Theorem 2.3.2. *If for a tidy evaluation plan q the corresponding path in automaton A^{tidy} ends with a solid (dashed, dash-dotted, dotted) edge then $q : ord, nodup$ ($q : ord, \neg nodup$, $q : \neg ord, nodup$, $q : \neg ord, \neg nodup$).*

Proof. For each transition from state s_1 to state s_2, labeled with axis a, it can be shown that if the properties in s_1 and *ord* and *nodup* hold for an evaluation plan q then it can be derived with the reasoning rules presented in Section 2.1 and Section 2.2 that for $q; a$ the properties *ord* and *nodup* hold or not, as is indicated by the type of the edge that represents the transition. If this was the last transition for a tidy evaluation plan $q'; \sigma; \delta; a$ then by Theorem 2.3.1 it follows that the properties in s_1 hold for q', and since they are set properties also for $q'; \sigma; \delta$. Moreover, since it trivially holds that *ord* and *nodup* hold for this evaluation plan it follows that the type of the edge indeed correctly indicates whether *ord* and *nodup* hold for $q'; \sigma; \delta; a$. ☐

Theorem 2.3.3. *The rules presented in Section 2.1 are complete for deriving the ord and nodup property for tidy evaluation plans.*

Proof. In A^{tidy} there is in each state a transition for each axis. Thus the automaton is indeed complete for deciding *ord* and *nodup* and therefore also the rules in Section 2.1 and Section 2.2. Moreover, the rules in Section 2.2 only derive negative set properties, and since none of the rules that derive positive set properties, *ord* and *nodup* depend upon negative set properties it follow that these do not lead to extra derivations of *ord* and *nodup*. ☐

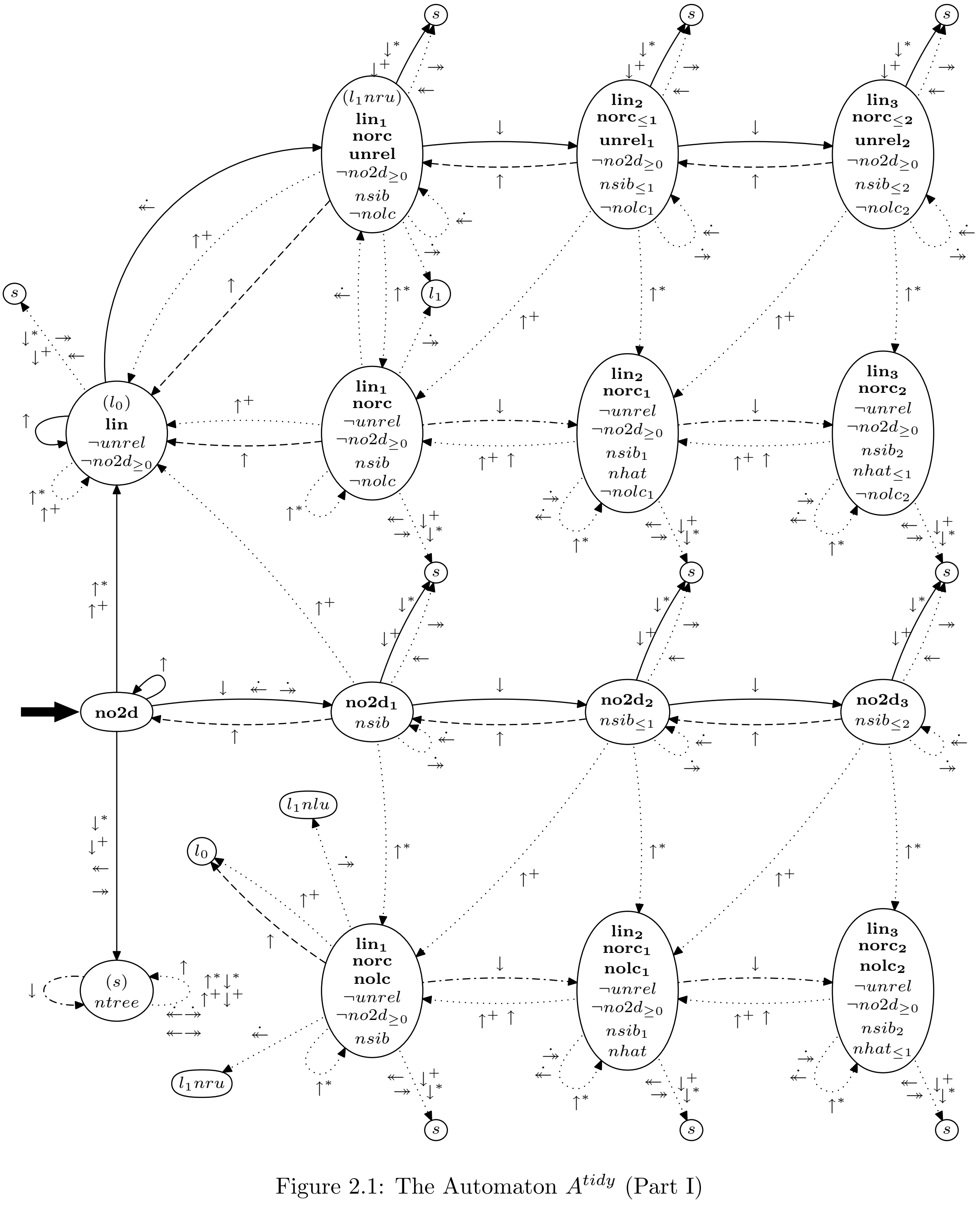


Figure 2.1: The Automaton A^{tidy} (Part I)

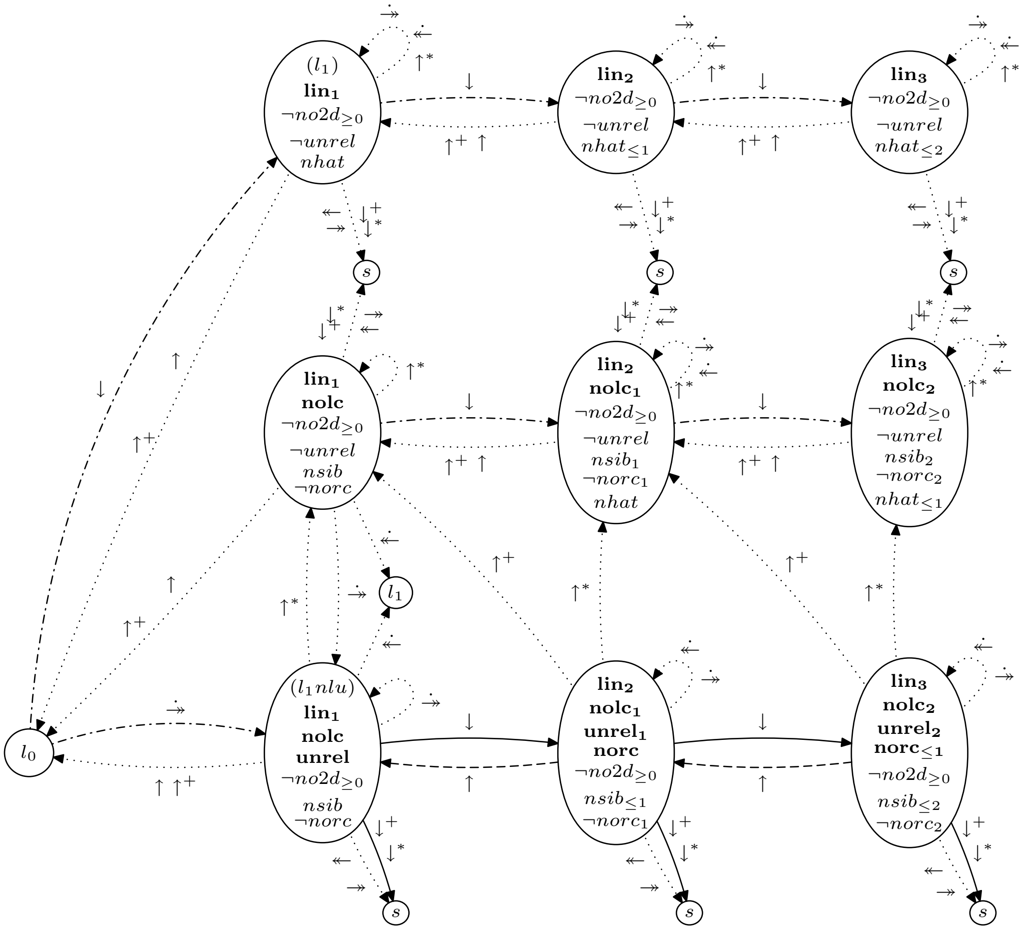


Figure 2.2: The Automaton A^{tidy} (Part II)

Chapter 3

Sloppy Evaluation Plans

3.1 Soundness and Completeness

3.1.1 Additional Inference Rules

Rules for ord

$$\frac{q : \mathit{ord}, \mathit{lin}}{q : \mathit{ord}_{\geq 0}} \ \textsc{ord-lin}$$

$$\frac{q : \mathit{ord}, \mathit{gen}}{q : \mathit{ord}_{\geq 0}} \ \textsc{ord-gen}$$

$$\frac{\begin{array}{l} q : \mathit{ord}_n \qquad n \geq 1 \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : \mathit{ord}_n} \ \textsc{ord-sib}$$

$$\frac{q : \mathit{ord}_n \qquad n \geq 1}{q; \uparrow : \mathit{ord}_{n-1}} \ \textsc{ord-prn}$$

$$\frac{q : \mathit{ord}_n \qquad n \geq 0}{q; \downarrow : \mathit{ord}_{n+1}} \ \textsc{ord-chl}$$

Rules for $\neg \mathit{ord}$

$$\frac{q : \mathit{ord}_n, \mathit{ntree} \qquad n \geq 0}{q : \neg \mathit{ord}_{\geq n+1}} \ \textsc{not-ord-ntree-gt}$$

$$\frac{q : \mathit{ord}_n, \mathit{ntree} \qquad n \geq 1}{q : \neg \mathit{ord}_{\leq n-1}} \ \textsc{not-ord-ntree-lt}$$

$$\frac{\begin{array}{l} q : \mathit{nsib} \\ a \in \{\uparrow^+, \uparrow^*, \twoheadrightarrow, \twoheadleftarrow\} \end{array}}{q; a : \neg \mathit{ord}_{\geq 0}} \ \textsc{not-ord-nsib-fpa}$$

$$\frac{\begin{array}{l} q : \neg \mathit{unrel} \\ a \in \{\downarrow^*, \downarrow^+, \twoheadrightarrow, \twoheadleftarrow\} \end{array}}{q : \neg \mathit{ord}_{\geq 0}} \ \textsc{not-ord-unrel-fpd}$$

$$\frac{\begin{array}{l} q : \neg \mathit{ord}_n \qquad n > 0 \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}, \twoheadrightarrow, \twoheadleftarrow\} \end{array}}{q : \neg \mathit{ord}_n} \ \textsc{not-ord-fpsib}$$

$$\frac{\begin{array}{l} q : \neg \mathit{nodup} \\ a \in \{\downarrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : \neg \mathit{ord}} \ \textsc{not-ord-nodup-chlsib}$$

$$\frac{\begin{array}{l} q : \neg \mathit{nodup} \\ a \in \{\twoheadleftarrow, \twoheadrightarrow, \downarrow^*, \downarrow^+, \uparrow^*, \uparrow^+\} \end{array}}{q; a : \neg \mathit{ord}_{\geq 0}} \ \textsc{not-ord-nodup-rec}$$

$$\frac{\begin{array}{l} q : \neg \mathit{ord}_n \qquad n \geq 1 \\ a \in \{\uparrow, \uparrow^*, \uparrow^+\} \end{array}}{q; a : \neg \mathit{ord}_{n-1}} \ \textsc{not-ord-up}$$

$$\frac{\begin{array}{l} q : \neg \mathit{ord}_{\leq n} \qquad n \geq 0 \\ a \in \{\downarrow, \downarrow^*, \downarrow^+\} \end{array}}{q; a : \neg \mathit{ord}_{\leq n+1}} \ \textsc{not-ord-down}$$

$$\frac{\begin{array}{l} q : \neg \mathit{no2d} \\ a \in \{\downarrow^*, \downarrow^+, \uparrow^*, \uparrow^+, \twoheadleftarrow, \twoheadrightarrow\} \end{array}}{q; a : \neg \mathit{ord}_{\geq 1}} \ \textsc{not-ord-no2d-rec}$$

Rules for $\neg nodup$

$$\frac{q : \neg nodup \qquad a \in A}{q; a : \neg nodup}\ \textsc{not-nodup-step}$$

Rules for gen

$$\frac{q : no2d}{q : gen}\ \textsc{gen-no2d} \qquad\qquad \frac{q : gen \qquad a \in \{\downarrow, \uparrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q : gen}\ \textsc{gen-step}$$

Rules for $unrel$

$$\frac{q : gen}{q : unrel}\ \textsc{unrel-gen}$$

Rules for $\neg no2d$

$$\frac{q : \neg unrel}{q : \neg no2d}\ \textsc{not-no2d-unrel}$$

3.1.2 Proofs for the Additional Rules

Lemma 3.1.1 (Unordered related nodes). *If an input sequence S contains two related nodes n_1 and n_2 then evaluating the parent axis for this sequence will again yield two unrelated, unordered nodes.*

Rules for ord

ORD-LIN (3.1.1)

$$\frac{q : ord, lin}{q : ord_{\geq 0}}$$

Proof. From the soundness rules of the tidy automaton we know that if $q : lin_n$ $(n \geq 0)$ holds, also $q : lin_{n+1}$ holds (LIN-UP). Additionally, we know that if $q : lin$ the also $q : norc$ (NORC-LIN). From these two rules we know that if $q : lin$ is true then also $q : norc_{\geq 0}$ holds. Finally the rule ORD-NORC-PRN shows that if a query q has the the *norc* and *ord* properties, then $q; \uparrow : ord$ also holds. So, since both *ord* and *norc* are preserved by the $\uparrow$ axis, we know that $q : ord_{\geq 0}$. ☐

ORD-GEN (3.1.1)

$$\frac{q : ord, gen}{q : ord_{\geq 0}}$$

Proof. The *gen* property states that every node in the result of a path expression has the same distance to the root. Clearly, this property is preserved after following the $\uparrow$ axis (i.e., if $q : gen$, then $q : gen_{\geq 0}$). There are never related nodes in the result of an evaluation plan that has the *gen* property, since two related nodes have a different distance to the document root. From this we know that if $q : gen$ holds, then also $q : norc$ is true. From the rule ORD-NORC-PRN we now know that if q has both the *ord* and the *gen* property, $q; \uparrow$ also has these two properties. By consequence, q has the $ord_{\geq 0}$ property. ☐

ORD-SIB (3.1.1)

$$\frac{\begin{array}{ll} q : ord_n & n \geq 1 \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} & \end{array}}{q; a : ord_n}$$

Proof. Following the $\dot{\twoheadrightarrow}$ or $\dot{\twoheadleftarrow}$ axis from a sequence results in a sequence, containing groups of child node from the same parent. Therefore, following the parent axis from that sequence will yield the same sequence as following it from the original sequence. The only two differences are that

1. the result sequence may contain duplicate nodes if there is more than one sibling of the same node in the input set, and
2. the result sequence does not contain the parent node of those nodes in the input sequence that do not have following/preceding siblings.

The order of the nodes in both sequences is the same and thus, the order of the output after following the parent axis any number of times will also be preserved. ☐

ORD-PRN (3.1.1)

$$\frac{q : ord_n \qquad n \geq 1}{q; \uparrow : ord_{n-1}}$$

Proof. This holds by definition 1.2.10. ☐

ORD-CHL (3.1.1)

$$\frac{\begin{array}{c} q : ord_n \\ n \geq 0 \end{array}}{q; \downarrow : ord_{n+1}} \text{ ORD-CHL}$$

Proof. For any query q, any document D and a node n in D, $[\![q; \downarrow; \uparrow]\!]_D(n)$ will produce a sequence with the following properties:

- Every node in $[\![q]\!]_D(n)$ occurs exactly k times in $[\![q; \downarrow; \uparrow]\!]_D(n)$ where k is the amount of its children;
- All duplicate occurrences of a node in $[\![q; \downarrow; \uparrow]\!]_D(n)$ are grouped; i.e., the order of the node remains unchanged;

This implies that if q has the *ord* property, so will $q; \downarrow; \uparrow$, and thus by definition 1.2.10, $q; \downarrow$ has the ord_1 property.
Suppose that a query q has the ord_n property. We know that $[\![q; \downarrow; \uparrow]\!]_D(n)$ contains a subset of the nodes in $[\![q]\!]_D(n)$, in the same order, possibly with duplicates. Since every node has at most one parent, the sequence $[\![q; \downarrow; \uparrow; \uparrow^n]\!]_D(n)$ also contains a subset of the nodes in $[\![q; \uparrow^n]\!]_D(n)$ in the same order, possibly with duplicates (prn^n stands for applying the $\uparrow$ axis n times). And thus, $[\![q; \downarrow;]\!]_D(n)$ has the ord_{n+1} property. ☐

Rules for $\neg ord$

NOT-ORD-NTREE-GT (3.1.1)

$$\frac{q : ord_n, ntree \qquad n \geq 0}{q : \neg ord_{\geq n+1}}$$

Proof. Since *ntree* holds for q, the rules NHAT-NTREE and NOT-NORC-NHAT allow us to deduce that $q : \neg norc_{\geq 0}$. So, $q; \upharpoonright^n$ has both the *ord* and the $\neg norc$ property and thus $q; \upharpoonright^n$ has the $\neg ord_1$ property. The result now contains two related nodes that are out of document order and lemma 3.1.1 says that if this will remain the case after following any amount of parent axes. ☐

NOT-ORD-NTREE-LT (3.1.1)

$$\frac{q : ord_n, ntree \qquad n \geq 1}{q : \neg ord_{\leq n-1}}$$

Proof. Same reasoning as for NOT-ORD-NTREE-GT. ☐

NOT-ORD-NSIB-FPA (3.1.1)

$$\frac{\begin{array}{l} q : nsib \\ a \in \{\upharpoonright^+, \upharpoonright^*, \twoheadrightarrow, \twoheadleftarrow\} \end{array}}{q; a : \neg ord_{\geq 0}}$$

Proof. Let q be a query and D a document with n, n_1 and n_2 nodes in D. If $q : nsib$ then let n_1 and n_2 be two siblings in $[\![q]\!]_D(n)$. It is easy to see that in following the $\upharpoonright^+$ axis, first all ancestors of n_1 will occur in the result, which are then followed by all ancestors of n_2. Since the intersection of both sets of ancestors is not empty, the result can never be in document order. A similar reasoning can be used for the $\upharpoonright^*$, $\twoheadrightarrow$ and $\twoheadleftarrow$ axes. ☐

NOT-ORD-UNREL-FPD (3.1.1)

$$\frac{\begin{array}{l} q : \neg unrel \\ a \in \{\downharpoonleft^*, \downharpoonleft^+, \twoheadrightarrow, \twoheadleftarrow\} \end{array}}{q : \neg ord_{\geq 0}}$$

Proof. Let q be a query and D a document with n, n_1 and n_2 nodes in D. If $q : \neg unrel$ then let n_1 and n_2 be two related nodes in $[\![q]\!]_D(n)$. It is easy to see that in following the $\downharpoonleft^+$axis, first all descendants of n_1 will occur in the result, which are then followed by all descendants of n_2. Since the intersection of both sets of descendants is not empty (because the two nodes are related), the result can never be in document order. A similar reasoning can be used for the $\downharpoonleft^*$, $\twoheadrightarrow$ and $\twoheadleftarrow$ axes. ☐

NOT-ORD-FPSIB (3.1.1)

$$\frac{\begin{array}{l} q : \neg ord_n \qquad n > 0 \\ a \in \{\dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}, \twoheadrightarrow, \twoheadleftarrow\} \end{array}}{q : \neg ord_n}$$

Proof. Let q be a query and D a document with n, n_1 and n_2 nodes in D. If $q : \neg ord_n$ then let n_1 and n_2 be nodes in $[\![q]\!]_D(n)$ whose n^th parents are out of document order. Since siblings have the exact same set of ancestors, also the n^th parents of those will be out of document order (i.e. $q; \dot{\twoheadrightarrow} : \neg ord_n$ and $q; \dot{\twoheadleftarrow} : \neg ord_n$). Since the $\dot{\twoheadrightarrow}$ and $\dot{\twoheadleftarrow}$ axes produce subsets of the $\twoheadrightarrow$ and $\twoheadleftarrow$ axes, this reasoning also holds for the latter. ☐

NOT-ORD-NODUP-CHLSIB (3.1.1)

$$\frac{\begin{array}{l} q : \neg nodup \\ a \in \{\downharpoonleft, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\} \end{array}}{q; a : \neg ord}$$

Proof. Repetitions of sequences are never in document order, if the sequences have more than one different node. ☐

NOT-ORD-NODUP-REC (3.1.1)

$$\frac{\begin{array}{l} q : \neg nodup \\ a \in \{\twoheadleftarrow, \twoheadrightarrow, \downharpoonleft^*, \downharpoonleft^+, \upharpoonleft^*, \upharpoonleft^+\} \end{array}}{q; a : \neg ord_{\geq 0}}$$

Proof. Since following any of the axes in $\{\twoheadleftarrow, \twoheadrightarrow, \downharpoonleft^*, \downharpoonleft^+, \upharpoonleft^*, \upharpoonleft^+\}$ more than once from the same node will produce an arbitrarily long repetition of the same sequence, there will exist no n for which the result of $q; \upharpoonleft^n$ is in document order. ☐

NOT-ORD-UP (3.1.1)

$$\frac{\begin{array}{l} q : \neg ord_n \qquad n \geq 1 \\ a \in \{\upharpoonleft, \upharpoonleft^*, \upharpoonleft^+\} \end{array}}{q; a : \neg ord_{n-1}}$$

Proof. For the $\upharpoonleft$ axis the soundness of this rule follwos from the definition 1.2.10 of indexed properties. Since the parent axis produces a subset of both the $\upharpoonleft^+$ and $\upharpoonleft^*$ axes, we can extend this rule with those. ☐

NOT-ORD-DOWN (3.1.1)

$$\frac{\begin{array}{l} q : \neg ord_{\leq n} \qquad n \geq 0 \\ a \in \{\downharpoonleft, \downharpoonleft^*, \downharpoonleft^+\} \end{array}}{q; a : \neg ord_{\leq n+1}}$$

Proof. For the $\downharpoonleft$ axis, this rule holds by definition 1.2.10. Since the $\upharpoonleft^*$ and $\downharpoonleft^*$ axis produce a superset of the $\downharpoonleft$ axis it also holds for those two axes. ☐

NOT-ORD-NO2D-REC (3.1.1)

$$\frac{\begin{array}{l} q : \neg no2d \\ a \in \{\downharpoonleft^*, \downharpoonleft^+, \upharpoonleft^*, \upharpoonleft^+, \twoheadleftarrow, \twoheadrightarrow\} \end{array}}{q; a : \neg ord_{\geq 1}}$$

Proof. **[For $\downharpoonleft^+$, $\downharpoonleft^*$, $\twoheadrightarrow$, $\twoheadleftarrow$]** After following any one of these axes, rule says that the *ntree* property holds. The rules NHAT-NTREE and NOT-NORC-NHAT allow us to deduce that $q : \neg norc_{\geq 0}$. So, $q; \upharpoonleft^n$ has both the *ord* and the $\neg norc$ property and thus $q; \upharpoonleft^n$ has the $\neg ord_1$ property. The result now contains two related nodes that are out of document order and lemma 3.1.1 says that if this will remain the case after following any amount of parent axes.
[For $\upharpoonleft^+$, $\upharpoonleft^*$] The sequences that result from following these axes from two different nodes share an arbitrarily long subsequence of nodes. The result sequence thus contains two identical subsequences of arbitrary length, which implies that there can be two related nodes that are not in document order, no matter how many times you follow the parent axis. ☐

Rules for $\neg nodup$

NOT-NODUP-STEP (3.1.1)

$$\frac{q : \neg nodup \qquad a \in A}{q; a : \neg nodup}$$

Proof. Axes evaluted more than once for the same context node produce duplicate results. ☐

Rules for *gen*

GEN-NO2D (3.1.1)

$$\frac{q : no2d}{q : gen}$$

Proof. Identical nodes always belong to the same generation. ☐

GEN-STEP (3.1.1)

$$\frac{q : gen \quad a \in \{\downarrow, \uparrow, \dot{\twoheadleftarrow}, \dot{\twoheadrightarrow}\}}{q : gen}\ \text{GEN-STEP}$$

Proof. Obviously, following the axes ↓, ↑, $\dot{\twoheadleftarrow}$ or $\dot{\twoheadrightarrow}$ from a set of context nodes that have the same distance to the root results in a new set of node that all have the same distance to the root. ☐

Rules for *unrel*

UNREL-GEN (3.1.1)

$$\frac{q : gen}{q : unrel}$$

Proof. Nodes that have the same distance to the root cannot be related. ☐

Rules for $\neg no2d$

NOT-NO2D-UNREL (3.1.1)

$$\frac{q : \neg unrel}{q : \neg no2d}\ \text{NOT-NO2D-UNREL}$$

Proof. If two nodes have a different distance to the root then it is impossible that the two nodes in fact denote the same node. ☐

3.2 Automata for Sloppy Evaluation Plans

3.2.1 The A^{sloppy}_{ord} Automaton

The rules in $\mathcal{R}$ allow us to construct a deterministic automaton that decides whether or not the result of a sloppy evaluation plan can contain duplicates and/or be out of document order. To indicate that the algorithm can be easily implemented, we consider two separate automata: one for deriving the *nodup* property (A^{sloppy}_{nodup}) and one for deriving the ord_0 property (A^{sloppy}_{ord}). Both automata accept expressions p that have the *ord* (*nodup*) property, in a time that is linear to the length of p; i.e., the number of step expressions in p.

In this infinite automaton (see Figures 3.2 and 3.3) accept states are indicated by a double border. Each state is labeled with the properties that hold in that state. The three-dot symbols indicate that the automaton has an infinite number of subsequent states with transitions from and to it that are the same as those of the last state before the symbol. The states are labeled with the same properties unless that property has an index. In this case, the index ascends in the subsequent states.

Note that the prefix of a path that has the *ord* property does not necessarily have the *ord* property itself; i.e., it is possible to return from an unordered state back into an ordered one.

3.2.2 The A^{sloppy}_{nodup} Automaton

NB: We actually need to check this ...

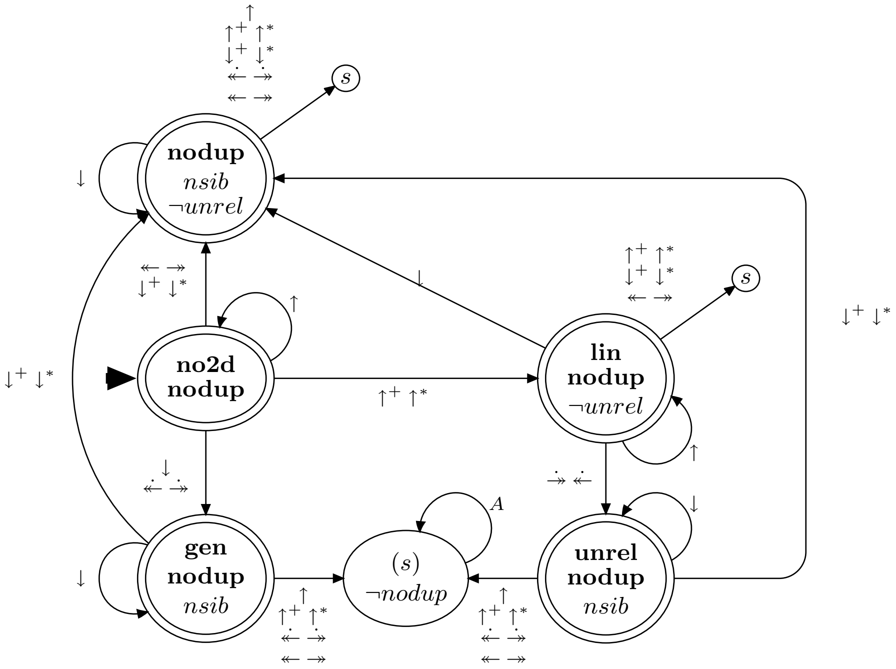


Figure 3.1: The A^{sloppy}_{nodup} Automaton

This finite automaton (see Figure 3.1) shows that, unlike the *ord* property, once the *nodup* property is lost, it never recurs; i.e., if a sloppy path evaluation plan q has the *nodup* property, then so will any sloppy path evaluation plan that has q as a prefix.

3.2.3 Proving Soundness and Completeness

Theorem 3.2.1. *A^{sloppy}_{ord} is sound for the ord property; i.e., A^{sloppy}_{ord} accepts only sloppy evaluation plans that have the ord property.*

Proof. For each transition from state s_1 to state s_2, labeled with axis a, it holds that there is a set of inference rules in $\mathcal{R}$ that justifies it; i.e., for every property that holds in s_2 the inference rules in $\mathcal{R}$ derive this property for a. Soundness now follows from the soundness of $\mathcal{R}$. ☐

Theorem 3.2.2. *A^{sloppy}_{ord} is complete for the ord property; i.e., every sloppy evaluation plan that has the ord property is accepted by A^{sloppy}_{ord}.*

Proof. ... ☐

Theorem 3.2.3. *A^{sloppy}_{nodup} is sound for the nodup property. ; i.e., A^{sloppy}_{nodup} accepts only sloppy evaluation plans that have the nodup property.*

Proof. Analogous to proof of Theorem 3.2.1. ☐

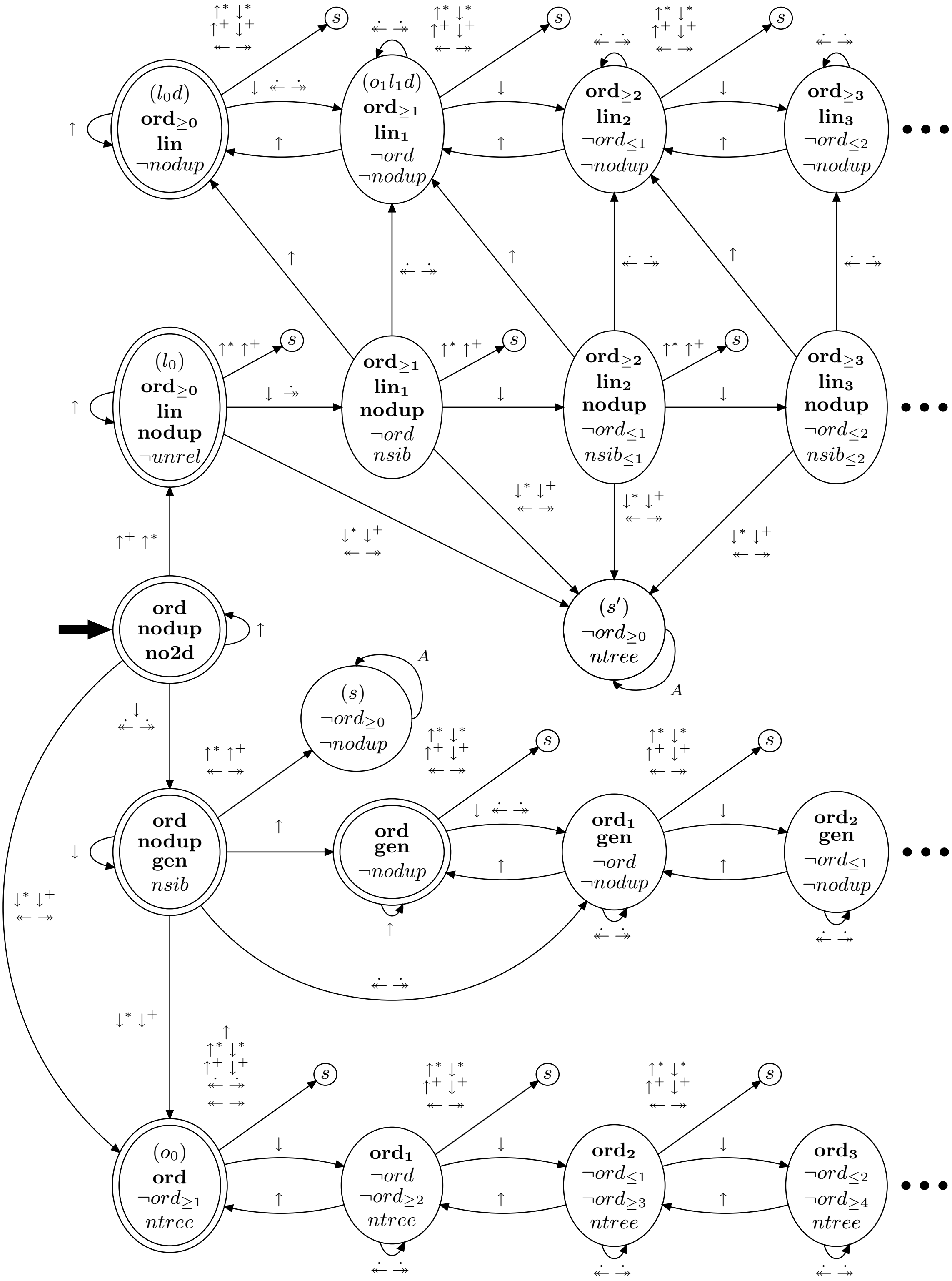


Figure 3.2: The A_{ord}^{sloppy} Automaton (Part I)

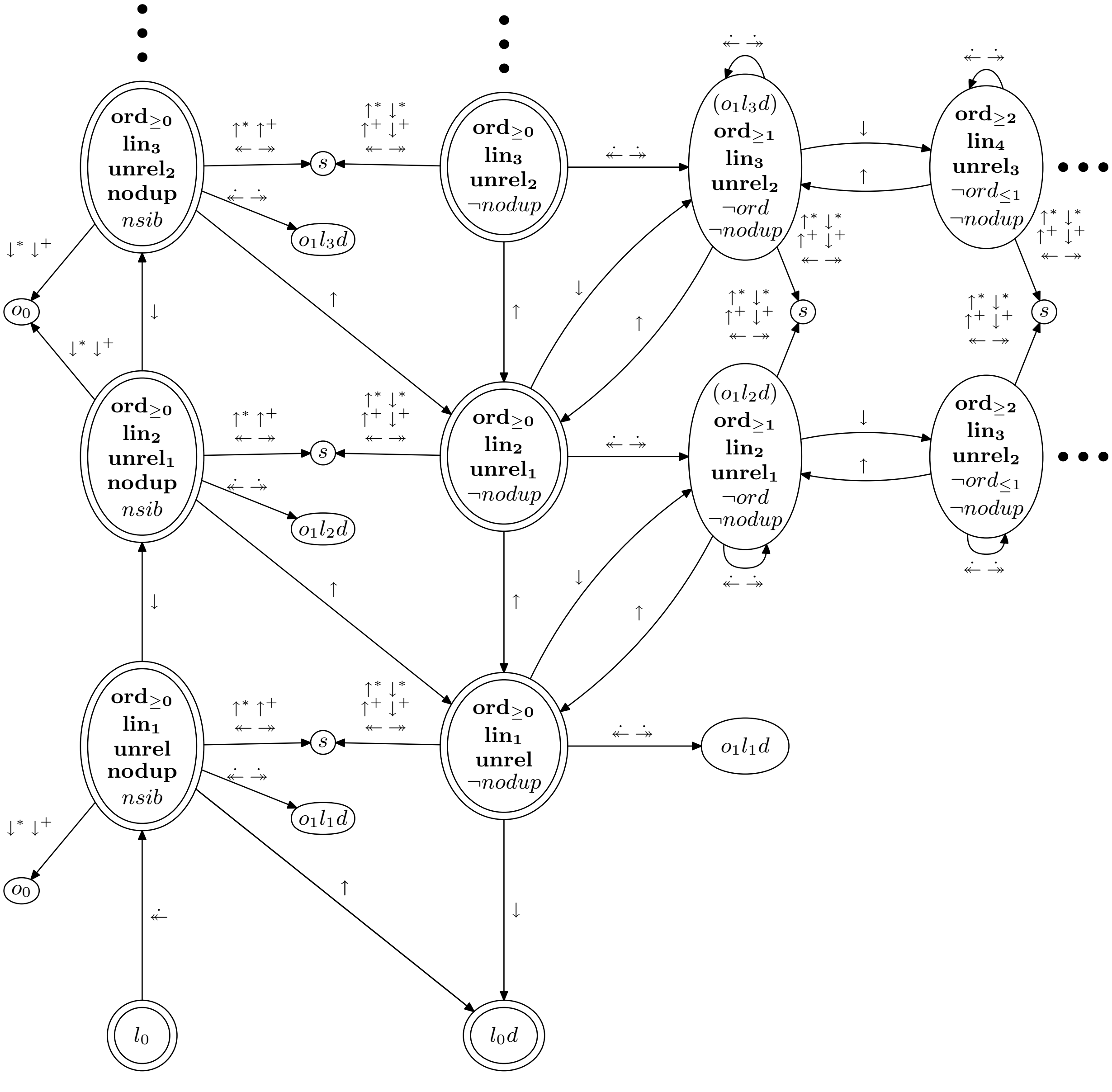


Figure 3.3: The A_{ord}^{sloppy} Automaton (Part II)

Chapter 4

Implementation

In this section, we describe how the automaton in Chapter 2.3 is implemented in the Galax XQuery engine [7].

4.1 Galax Architecture

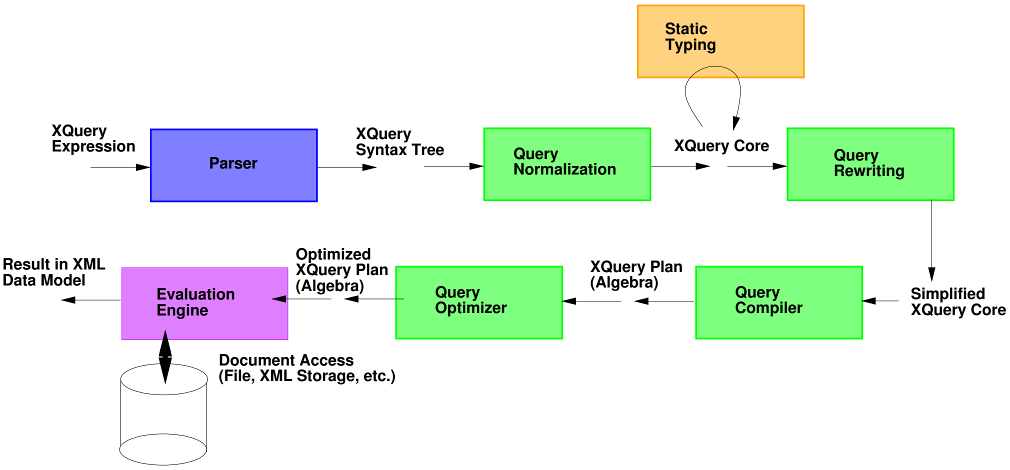


Figure 4.1: Galax Compilation Architecture

Galax [7] is an implementation of the family of XQuery 1.0 specifications [3] designed for completeness and conformance with the W3C standard. To achieve these goals, its architecture parallels the processing model described in the XQuery Formal Semantics [5]. Figure 4.1 depicts the complete Galax architecture. The upper half of the picture corresponds to the processes described in the XQuery Formal Semantics, and the lower half corresponds to a typical query compiler, which is not discussed here.

In Galax, the query is first parsed to produce an abstract syntax tree (AST). This AST is then *normalized* into an equivalent expression in the XQuery Core language [5], which is a simpler subset of the complete language. Normalization serves two purposes. First, it makes the implicit semantics of XQuery expressions explicit by expanding every expression (e.g., existential quantification in predicates, casting of arithmetic operands, etc.). Second, normalization simplifies subsequent compilation steps (e.g., static typing and rewriting), because those steps operate on the small, simpler Core language instead of the complete language.

To illustrate, here is the actual normalized expression for the expression `$sur/procedure/incision/..` in Section 1.1:

```
fs:distinct-docorder(
  let $fs:sequence :=
    fs:distinct-docorder(
      let $fs:sequence :=
        fs:distinct-docorder(
          let $fs:sequence := $sur return
          let $fs:last := count($fs:sequence) return
          for $fs:dot at $fs:position in $fs:sequence
          return child::procedure)
      return
      let $fs:last := count($fs:sequence) return
      for $fs:dot at $fs:position in $fs:sequence
      return child::incision)
  return
  let $fs:last := count($fs:sequence) return
  for $fs:dot at $fs:position in $fs:sequence
  return parent::node())
```

The normalized expression is extremely tidy: It sorts by document order at every step, ensuring a proper semantics for the query. It also binds variables that model the implicit *context* of each step in a path expression: `$fs:sequence` denotes the context sequence, `$fs:dot` denotes the context node, `$fs:last` denotes the length of the context sequence, and `$fs:position` denotes the position of the context node in the context sequence. Because normalization occurs top-down, these variables are bound even if they are not used in subsequent expressions. The rewriting phase (last step in upper half of Figure 4.1), which follows static typing, removes unused variables.

The static-typing phase takes a Core expression and, if the expression is well typed, annotates each sub-expression with its inferred type. The `DDO` optimization uses type annotations to infer the *maxone* property, which is required by the automaton's start state. The *maxone* property is easily derived from the inferred types. For example, the `$sur` variable and all variables bound in a `for` iteration (e.g., `$fs:dot`) have the *maxone* property, because their type is always a single item.

In Galax, static typing is the preferred method for deriving the *maxone* property. Static typing, however, is an optional feature of XQuery, and other implementations may prefer a simpler technique to derive the *maxone* property. If static typing is disabled, we use a variant of XQuery's static typing that we call *weak typing*. With weak typing, each sub-type check, which might fail statically, is replaced by a type coercion, which never fails statically. Sub-type checks occur in the semantics of function calls, built-in operators, and many other expressions. For example, the type assertion in the following `let` expression requires that the type of *Expr* be a sub-type of `element(surgery)`. If the sub-type check succeeds, then the type of `$sur` is guaranteed to be one surgery element.

```
let $sur as element(surgery) := Expr
return $sur/procedure/incision/..
```

Weak typing replaces the type assertion by a type coercion (the `treat as` expression), which does not require a static sub-type check. Instead, we can immediately infer that the type of `$sur` is `element(surgery)`:

```
let $sur := Expr treat as element(surgery)
return $sur/procedure/incision/..
```

Weak typing is easy to implement, because it does not require sub-type checking, and it has the important property that if an expression does not fail at run time, then the value of the expression is guaranteed to have the type to which the expression was coerced.

Type annotations are illustrated below on our example.

```
fs:distinct-docorder(
 for $fs:dot [element incision] in
  fs:distinct-docorder(
   for $fs:dot [element procedure] in
    fs:distinct-docorder(
     for $fs:dot in $sur  [element surgery]
     return child::procedure [element procedure*]
    ) [element procedure*]
   return child::incision [element incision]
  ) [element incision*]
 return parent::node() [element]
) [element*]
```

The rewriting phase follows static typing. This phase takes as input a Core expression, a set of rewriting rules, and applies the rules recursively to each sub-expression until it reaches a fixed point. Current rewriting rules include: type-based optimizations to eliminate dynamic type checks; conversion of dynamically dispatched operators to statically dispatched; removal of unused variables, and others [4]. After eliminating unused variables, our example expression is:

```
fs:distinct-docorder(
  for $fs:dot in
    fs:distinct-docorder(
      for $fs:dot in
        fs:distinct-docorder(
          for $fs:dot in $sur return
          child::procedure)
      return child::incision)
  return parent::node())
```

4.2 Applying the `DDO` Optimization

The `DDO` optimization described in Chapter 2 is applied as part of the rewriting phase. It is implemented in three steps:

1. Since the automaton operates on the path-expression fragment of XQuery, the first step identifies path expressions by applying pattern matching during a top-down traversal of the core AST.
2. The automaton is applied to each path expression identified in the first step, and each step expression is annotated with the *ord* and *nodup* properties inferred during the application of the automaton.
3. The last step applies a rewrite rule that removes redundant `ddo` operations or replaces them with `fs:docorder` or `fs:distinct` operators, using the annotations derived in the second step. The rewrite rule is applied by the general Galax rewriting phase, in conjunction with other rewrite rules.

Here is the core expression with the annotations inferred by the automaton:

```
fs:distinct-docorder(
  for $fs:dot in
    fs:distinct-docorder(
      for $fs:dot in
        fs:distinct-docorder(
          for $fs:dot in $sur
          return child::procedure [nodup,ord])
      return child::incision [nodup,ord])
  return parent::node() [ord])
```

And here is the final optimized expression:

```
fs:distinct(
  for $fs:dot in
   for $fs:dot in
     for $fs:dot in $sur return child::procedure
   return child::incision
  return parent::node())
```

4.3 Evaluating Core XQuery

The core XQuery expression which results from normalization can be seen as a simple kind of a query plan. In fact, Galax currently evaluates such core queries in a very literal way, in a top-down fashion. In the context which interests us, the most important operation is `fs:distinct-docorder`, which is a special built-in function[1] that sorts nodes in document order and removes duplicate nodes. In Galax, this operation is implemented as a merge sort followed by a linear duplicate removal on the sorted list.

However efficient the implementation of sorting might be, applying it numerous times to large collections will degrade performances. Moreover, this is a blocking operation. Its evaluation requires to materialize all of the nodes in memory and prevents the use of a pipelined evaluation for the XPath expression. Note that we use of normalized expressions as a convenient way to express the more general problem related to the need of sorting by document order and duplicate removal. It is indeed not specific to Galax as any implementation would have to decide when it is necessary to perform sorting and if so, sorting would prevent the use of pipelined query plans.

---

[1] In the `fs` namespace, for “Formal Semantics”.

Chapter 5

The `DDO` Optimization in Context

So far, we have described the `DDO` optimization and explained its implementation in isolation from any other optimization. In practice, its benefits can be maximized by doing some appropriate preparatory work on the query, and by exploiting the result of `DDO` optimization in further query processing steps. In this section, we explore possible preparation steps, and advanced optimizations enabled by the `DDO` approach.

5.1 Preparing the `DDO` Optimization

One of the most typical problems, as we have seen from the automaton in Chapter 2, is that applications of the descendant axis (used notably in the `//` operator) will lead to a sink state. As a direct consequence, the algorithm will not be able to remove most of the `ddo` operations beyond the first `descendant` step. Consider for instance the following simple path expression which uses both XPath predicates and the descendant axis.

```
$sur//procedure[1]/anesthesia
```

Applying the `DDO` optimization on that query directly results in the following optimized core expression.

```
fs:docorder(
  for $fs:dot in (
    fs:docorder(
      for $fs:dot in (
        for $fs:dot in $sur
        return descendant-or-self::node()
      )
      return
        for $fs:dot at $fs:position in
          child::procedure
        return
          if (op:equal($fs:position,1))
          then ($fs:dot)
          else ()
    )
  return child::anesthesia
)
```

As expected, only the two first sorting operations for the first two steps[1] can be removed.

The reason for this limitation is that the nodes selected after the application of a descendant axis can be related (e.g., through a descendant-ancestor relationship). As a result, we cannot infer

---

[1] Recall that `//a` is equivalent to `/descendant-or-self::node()/child::a`.

the *norc* or *unrel* properties in those cases. However, we may know from the the schema that there is no recursive type involved in the query [16]. This is the case for this query for the DTD given at the beginning of the paper. This enables us to rewrite the query using only child steps, as follows:

```
$sur/procedure[1]/anesthesia
```

This, in turn, allows the complete removal of all sorting by document order and duplicate removal operations. The main idea is that schema information can be used to rewrite the query into an equivalent one which uses only the axis for which the automaton algorithm is the most effective (notably children and parent).

5.1.1 Further Optimizations

The presence of sorting operations within the query plan prevents the use of some other important optimization techniques. After the `DDO` optimization is applied, some of those optimizations may be enabled again and provide additional gains. This is true of important standard optimizations such as loop-fusion or the use of pipe-lined evaluation.

Loop fusion is well known from database and programming languages optimizations [8], and eliminates the need for storing intermediate data. The left associativeness of XPath expressions naturally results in normalized queries that involve consecutive loops which materialize intermediate result sequences after each step. For instance, consider one our previous example query: `$sur/preocedure/incision/..`. After applying the `DDO` optimization, this results in the following core expression.

```
fs:distinct(
  for $fs:dot in
    for $fs:dot in
        for $fs:dot in $sur
        return child::procedure
    return child::incision
  return parent::node()
)
```

A naive evaluation strategy based for that expression would materialize the intermediate result sequences before iterating on the next step. However, in the absence of intermediate sorting operation, this expression can be rewritten using loop fusion to avoid materialization, as follows.

```
fs:distinct(
  for $fs:dot in $sur return
    for $fs:dot in child::procedure return
      for $fs:dot in child::incision return
        parent::node()
)
```

As we will see in the next section, this query plan performs significantly faster and with much less memory than the original query plan.

Chapter 6

Experimental Results

The `DDO` optimization is implemented in the development version of Galax, which can be accessed from Galax's public CVS repository[1]. This section reports on experimental results using the development implementation. The experiments consist of the XMark benchmark suite [19] applied to documents of various sizes. The XMark benchmark consists of twenty queries applied to a document consisting of auctions, bidders, and items. The queries exercise most of XQuery's features (selection, aggregation, grouping, joins, and element construction, etc.) and all contain at least one path expression.

6.1 Analysis of XMark Queries

Of the 82 path expressions in the XMark suite, 75 path expressions use only the child axis. From Section 2.3, we know that both the *ord* and *nodup* properties hold for every step in paths containing only child axes, and as expected, our algorithm indeed removes every `ddo` operation in these 75 path expressions.

The remaining seven path expressions (in Queries 6, 7, 14, and 19) each contain one descendant-or-self step due to the use of `//`. From the automaton in Section 2.3, we know that all child steps following a descendant-or-self axis require intervening sort operations. In these cases, our algorithm replaces each `ddo` operation by a `fs:docorder` (sort by document order) operation.

The XMark input documents have a corresponding DTD, which is used by Galax to infer the types of expressions during static analysis. As described in Section 5.1, type information can be used to rewrite each descendant-or-self step in into a sequence of child steps. This rewriting applies to all the descendant-or-self steps in XMark queries, which subsequently permits all `ddo` operations to be eliminated.

6.2 Performance of the `DDO` Optimization

We measured query evaluation time and total live-memory usage for each XMark query without optimization (normal) and with the `DDO` optimization (optimized). Query evaluation time excludes static analysis and document-loading time, which is negligible compared to query evaluation time. Our platform was an Intel Pentium 4, 2.26GHz CPU, 512 KB cache, with 512 MB main memory running Debian GNU/Linux. The input document of 20 MB was generated by the XMark document generator.

We partition the XMark queries into two groups: those that contain joins and therefore do not scale well with the size of the input document (Queries 8–12), and those that do not contain joins and do scale well (Queries 1–7,13–20). Because measurable improvements were more significant for the non-scalable queries than for the scalable ones, we focus on the non-scalable queries.

---

[1] `http://ncc.research.bell-labs.com:8081/cgi-bin/cvsweb/`

	XMark 10 MB		**XMark 20 MB**	
	normal	**optimized**	**normal**	**optimized**
Q01	0.249	0.277	0.532	0.481
Q02	0.344	0.376	0.867	0.861
Q03	0.795	0.777	1.663	1.558
Q04	0.716	0.654	1.405	1.310
Q05	0.278	0.260	0.566	0.511
Q08	354.447	327.039	1,462.012	1,339.132
Q09	436.658	404.055	1,809.613	1,648.991
Q10	59.106	46.186	231.376	175.301
Q11	913.394	785.322	3,742.743	3,205.567
Q12	257.275	231.528	1,044.357	949.338
Q13	0.780	0.910	2.428	2.310
Q15	0.216	0.164	0.480	0.369
Q16	0.292	0.598	0.598	0.419
Q17	0.392	0.310	0.837	0.653
Q20	2.178	1.888	4.319	3.907

Figure 6.1: Time results for the XMark queries on a 10 MB and a 20 MB document.

All optimized queries not containing the descendant-or-self axis have faster evaluation times than the non-optimized variants. The evaluation times of scalable queries range from 0.5 to 4.5 seconds, and the optimized variants run from 6–30% faster. The improvements here are modest, because the scalable queries tend to have high selectivity and correspondingly small intermediate results, therefore, removing the `ddo` operations does not have a large absolute impact.

The effect of the `DDO` optimization on non-scalable queries is more significant. Figure 6.2 shows query evaluation times and Figure 6.1 shows live-memory usage for the non-scalable queries. Query 10 shows the biggest improvement in evaluation time: 24%. These and other tests show that the relative improvement in evaluation time of the `DDO` optimization grows with the size of the input documents. For a 45 MB XMark document, Query 8 shows an improvement of 46% compared with an 8% on a 20MB document. Note that Galax evaluates normalized expressions top-down over an in-memory representation of input documents, and consequently, joins are implemented by computing cartesian products, which accounts for the long evaluation times of the non-scalable queries on a 20MB document.

All optimized queries have substantially smaller memory usage than their non-optimized variants. For scalable queries, memory usage is reduced from approximately 30–60%, and for non-scalable queries in Figure 6.2, the memory usage is reduced by 46–67%. In Galax, the `ddo` operation is implemented as a merge sort followed by linear-time duplicate elimination on the sorted sequence. The merge sort uses constant heap space and logarithmic stack space [17], so the time gained is not due to an inefficient implementation of the sorting algorithm. Instead, the improvement is due to the large number of `ddo` operations performed in the non-optimized variant. For example, on a 10 MB document, more than 2,550 `ddo` operations are applied in Query 8, and all these operations are eliminated by our algorithm.

6.3 Applying Further Optimizations

The previous experiments measured the effectiveness of the `DDO` optimization in isolation, but as discussed in Section 5, it is most effective when applied in the context of other optimizations.

To illustrate the interactions of optimizations, we applied the following query, which selects the names of all bidders in Belgiums in open auctions, to a XMark-generated document of 10 MB.

```
$auction//person[$auction//bidder/personref/@person= @id][address/country="Belgium"]/name
```

	XMark 10 MB		XMark 20 MB	
	normal	**optimized**	**normal**	**optimized**
Q01	245.695	125.609	730.654	294.008
Q02	989.738	983.199	2,113.598	1,929.492
Q03	236.551	188.414	579.125	395.426
Q04	118.449	60.660	346.219	149.648
Q05	75.715	50.102	213.926	107.555
Q08	197,015.215	71,633.125	1,010,620.578	334,070.398
Q09	236,302.141	86,914.245	1,207,978.199	404,084.164
Q10	55,463.738	35,728.940	211,343.633	113,640.184
Q11	603,942.090	261,674.828	3,053,431.836	1,278,016.062
Q12	121,213.586	56,694.789	602,205.074	274,118.262
Q13	7,234.121	7,233.789	11,880.789	11,868.188
Q15	366.684	130.457	1,095.237	480.664
Q16	72.609	27.746	196.609	77.164
Q17	624.906	429.962	1,464.176	921.242
Q20	1,397.039	626.840	3,442.734	1,545.578

Figure 6.2: Memory results for the XMark queries on a 10 MB and a 20 MB document.

Figure 6.3 shows the evaluation time and memory consumption for the query evaluated with schema-based optimization, the additional `DDO` optimization, and with loop fusion.

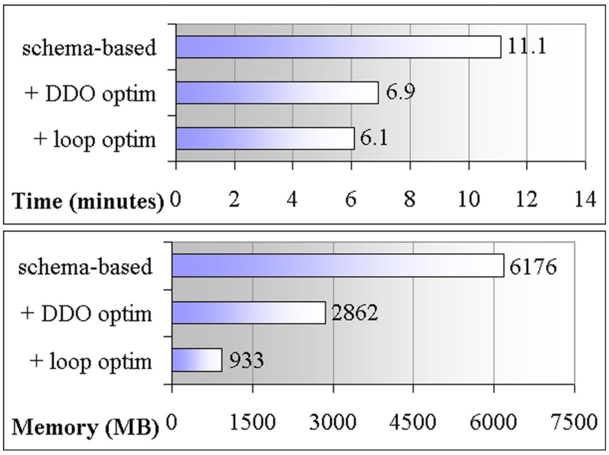


Figure 6.3: Performance of usecase query on 10 MB document

In addition to reducing evaluation time by one third, the `DDO` optimization reduces memory usage by more than half. The `DDO` optimization further enables application of loop fusion, which also reduces evaluation time and reduces memory usage by more than two-thirds. Further experiments are necessary to measure the general impact of loop fusion.

Chapter 7

Related Work and Discussion

Numerous papers address the semantics and efficient evaluation of XPath. Many address sorting and duplicate elimination, which is a strong indication of the importance of this problem. Duplicate elimination and avoiding sorting is particularly important in streaming evaluation strategies [18]. Helmer et al [13] present an evaluation technique that avoids the generation of duplicates, which is crucial for pipelining steps of a path expression. Grust [11] proposes a similar but more holistic approach, which uses a preorder and postorder numbering for XML documents to accelerate the evaluation of path location steps in XML-enabled relational databases. The preorder and postorder numbering of nodes can substantially accelerate the evaluation of several axes by using B-tree indices. In subsequent work, Grust [12] introduces the 'staircase join', a tree-aware operator that can further speed up XPath evaluation. By pruning the context list, the generation of duplicates and out-of-order nodes in the intermediate results is avoided, clearing the way for full pipelining. Most of this work however tightly contrains the supported language. In contrast, the techniques presented in this work support the entire XQuery language.

Other work considers the efficient evaluation of XPath in more general terms. In experimental results, Gottlob et al [9] show that naive implementations of XPath are unscalable. They present a bottom-up approach that uses context-value tables for XPath evaluation with complexity $\mathcal{O}(|D|^4 \times |Q|^2)$, where D the size of the data and Q the size of the query. They also present a top-down algorithm that applies axes and functions to context tuples in bulk, but that relies on very efficient evaluation of axes. In subsequent work [10], the complexity bounds are improved to $\mathcal{O}(|D|^2 \times |Q|^2)$, an important improvement since document size dominates. This work however avoids the confrontation with the problem of document order whereas our work focusses on it.

Our technique complements much of this research, and we believe is a necessary first step in any complete implementation of XPath. The `DDO` optimization applies to the entire XPath 2.0 language and produces semantically correct and simplified expressions that can be input to query planners that implement various evaluation strategies. Aside from that, the completeness of our approach ensures optimal results for a considerable part of the language.

In our own work, we will continue to improve logical rewritings early in the compilation pipeline as well as implement more sophisticated evaluation strategies later in the compilation pipeline. Currently, the `DDO` optimization is limited to path expressions, but we plan to extend the technique to all of XQuery. Simple improvements include propogating properties computed within a path expression to subsequent uses of the expression, e.g., `let`-bound variables, across function calls, etc., and using static typing properties (e.g., *maxone*) not just for the head of the path expression, but for intermediate steps. More difficult is determining the interaction of document order and duplicates with FLWOR expressions that include order-by clauses, the `unordered` expression, and aggregation functions.

In ongoing research, which is discussed in [14], we also plan to establish more efficient implementation strategies for XPath expressions. Unlike this work the approach is based on smart pipeline-enabling algorithms that are not based on the formal semantics.

Appendix A

Correctness of the A^{tidy} Automaton

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
no2d $[lin]$ $[lin_1]$ $[no2d_1]$	 LIN-NO2D LIN-UP NO2D-UP	$[no2d_1]$ $[ord]$ $[nodup]$ $[nsib]$	P-CHL ORD-NO2D-STEP NODUP-NO2D-STEP NSIB-CHLSIB	$[ord]$ $[nodup]$ $[ntree]$	ORD-NO2D-STEP NODUP-NO2D-STEP NTREE-SINK	$[ord]$ $[nodup]$ $[ntree]$	ORD-NO2D-STEP NODUP-NO2D-STEP NTREE-SINK	$[no2d_1]$ $[ord]$ $[nodup]$ $[nsib]$	P-SIB ORD-NO2D-STEP NODUP-NO2D-STEP NSIB-CHLSIB	$[ord]$ $[nodup]$ $[ntree]$	ORD-NO2D-STEP NODUP-NO2D-STEP NTREE-SINK
(s) $ntree$ $[nhat]$ $[nsib]$ $[\neg norc]$ $[\neg unrel]$ $[\neg no2d]$	 NHAT-NTREE NSIB-NHAT NOT-NORC-NHAT NOT-UNREL-NTREE NOT-NO2D-NSIB	$[nodup]$ $[ntree]$ $[\neg ord]$	NODUP-CHL NTREE-STEP NOT-ORD-UNREL-DOWN	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-STEP NOT-ORD-UNREL-DOWN NOT-NODUP-UNREL-DSC	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-STEP NOT-ORD-UNREL-DOWN NOT-NODUP-UNREL-DSC	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-STEP NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-STEP NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA
lin $\neg unrel$ $\neg no2d_{\geq 0}$ $[lin_1]$ $[nolc]$ $[norc]$	 LIN-UP NOLC-LIN NORC-LIN	$[lin_1]$ $[nodup]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nhat]$ $[\neg ord]$	P-CHL NODUP-CHL NOT-NO2D-STEP NOT-UNREL-PRNCHL NOT-UNREL-CHL NOT-ORD-UNREL-DOWN	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-UNREL-DOWN NOT-NODUP-UNREL-DSC	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-STEP NOT-ORD-UNREL-DOWN NOT-NODUP-UNREL-DSC	$[lin_1]$ $[nolc]$ $[unrel]$ $[nodup]$ $[\neg no2d_{\geq 0}]$ $[nsib]$ $[\neg norc]$ $[\neg ord]$	P-SIB NOLC-FLS UNREL-NOLC-FLS NODUP-LIN-PRNSIB NOT-NO2D-STEP NSIB-CHLSIB NOT-NORC-UNREL-FLS NOT-ORD-NSIB-SIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nhat$ $[lin_2]$ $[\neg nolc]$ $[\neg norc]$ $[nsib]$	 LIN-UP NOT-NOLC-NHAT NOT-NORC-NHAT NSIB-NHAT	$[lin_2]$ $[nodup]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nhat_{\leq 1}]$ $[\neg ord]$	P-CHL NODUP-CHL NOT-NO2D-STEP NOT-UNREL-PRNCHL NOT-UNREL-CHL NP-CHL NOT-ORD-UNREL-DOWN	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-UNREL-DOWN NOT-NODUP-UNREL-DSC	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-STEP NOT-ORD-UNREL-DOWN NOT-NODUP-UNREL-DSC	$[lin_1]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nhat]$ $[\neg ord]$ $[\neg nodup]$	P-SIB NOT-NO2D-STEP NOT-UNREL-NOLC-FLS NHAT-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_i}$ $(i > 1)$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nhat_{\leq i-1}$ $[lin_{i+1}]$ LIN-UP $[\neg nolc_{\leq i-1}]$ NOT-NOLC-NHAT $[\neg norc_{\leq i-1}]$ NOT-NORC-NHAT $[nsib_{\leq i-1}]$ NSIB-NHAT	$[lin_{i+1}]$ P-CHL $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nhat_{\leq i}]$ NOT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_i]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[nhat_{\leq i-1}]$ NHAT-SIB NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\mathbf{nolc}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib$ $\neg norc$	$[lin_2]$ P-CHL $[nolc_1]$ P-CHL $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib_1]$ NP-CHL $[\neg norc_1]$ NP-CHL $[nhat]$ NHAT-UNREL-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_1]$ P-SIB $[nolc]$ NOLC-FLS $[unrel]$ UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-SIB $[\neg norc]$ NOT-NORC-UNREL-FLS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{nolc_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_1$ $\neg norc_1$ $nhat$ $[\neg nolc]$ NOT-NOLC-NHAT $[\neg norc]$ NOT-NORC-NHAT $[nsib]$ NSIB-NHAT	$[lin_3]$ P-CHL $[nolc_2]$ P-CHL $[nodup]$ NODUP-CHL $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_2]$ NP-CHL $[\neg norc_2]$ NP-CHL $[nhat_1]$ NHAT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_2]$ P-SIB $[nolc_1]$ NOLC-FLS $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_1]$ NP-SIB $[\neg norc_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{nolc_{i-1}}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_{i-1}$ $\neg norc_{i-1}$ $nhat_{\leq i-2}$ $[\neg nolc_{\leq i-2}]$ NOT-NOLC-NHAT $[\neg norc_{\leq i-2}]$ NOT-NORC-NHAT $[nsib_{\leq i-2}]$ NSIB-NHAT	$[lin_{i+1}]$ P-CHL $[nolc_i]$ P-CHL $[nodup]$ NODUP-CHL $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_i]$ NP-CHL $[\neg norc_i]$ NP-CHL $[nhat_{\leq i-1}]$ NHAT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_i]$ P-SIB $[nolc_{i-1}]$ NOLC-FLS $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{i-1}]$ NP-SIB $[\neg norc_{i-1}]$ NP-SIB $[nhat_{\leq i-2}]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\mathbf{nolc}$ $\mathbf{unrel}$ $\neg no2d_{\geq 0}$ $nsib$ $\neg norc$	$[lin_2]$ P-CHL $[nolc_1]$ P-CHL $[norc]$ NORC-UNREL-CHL $[unrel_1]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-CHL $[\neg norc_1]$ NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[lin_1]$ P-SIB $[nolc]$ NOLC-FLS $[unrel]$ UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-CHLSIB $[\neg norc]$ NOT-NORC-FLS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{nolc_1}$ $\mathbf{norc}$ $\mathbf{unrel_1}$ $\neg no2d_{\geq 0}$ $nsib_{\leq 1}$ $\neg norc_1$ $[unrel]$ UNREL-DOWN	$[lin_3]$ P-CHL $[nolc_2]$ P-CHL $[norc_{\leq 1}]$ NORC-UNREL-CHL P-CHL $[unrel_2]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 2}]$ NSIB-CHLSIB NP-CHL $[\neg norc_2]$ NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[lin_2]$ P-SIB $[nolc_1]$ NOLC-FLS $[norc]$ NORC-UNREL1-SIB $[unrel_1]$ UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-SIB $[\neg norc_1]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{nolc_{i-1}}$ $\mathbf{norc_{\leq i-2}}$ $\mathbf{unrel_{i-1}}$ $\neg no2d_{\geq 0}$ $nsib_{\leq i-1}$ $\neg norc_{i-1}$ $[unrel_{\leq i-1}]$ UNREL-DOWN	$[lin_{i+1}]$ P-CHL $[nolc_i]$ P-CHL $[norc_{\leq i-1}]$ NORC-UNREL-CHL P-CHL $[unrel_i]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i}]$ NSIB-CHLSIB NP-CHL $[\neg norc_i]$ NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[lin_i]$ P-SIB $[nolc_{i-1}]$ NOLC-FLS $[norc_{\leq i-2}]$ NORC-UNREL1-SIB P-SIB $[unrel_{i-1}]$ UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i-1}]$ NSIB-CHLSIB NP-SIB $[\neg norc_{i-1}]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ **norc** **unrel** $\neg no2d_{\geq 0}$ $nsib$ $\neg nolc$	$[lin_2]$ P-CHL $[norc_{\leq 1}]$ NORC-UNREL-CHL P-CHL $[unrel_1]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-CHL $[\neg nolc_1]$ NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[lin_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[nhat]$ NHAT-NOLC-FLS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{norc_{\leq 1}}$ $\mathbf{unrel_1}$ $\neg no2d_{\geq 0}$ $nsib_{\leq 1}$ $\neg nolc_1$	$[lin_3]$ P-CHL $[norc_{\leq 2}]$ NORC-UNREL-CHL P-CHL $[unrel_2]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 2}]$ NSIB-CHLSIB NP-CHL $[\neg nolc_2]$ NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-STEP	$[lin_i]$ P-SIB $[norc_{\leq i-1}]$ NORC-UNREL1-SIB P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-SIB $[\neg nolc_1]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{norc_{\leq i-1}}$ $\mathbf{unrel_{i-1}}$ $\neg no2d_{\geq 0}$ $nsib_{\leq i-1}$ $\neg nolc_{i-1}$ $[unrel_{\leq i-1}]$ UNREL-DOWN	$[lin_{i+1}]$ P-CHL $[norc_{\leq i}]$ NORC-UNREL-CHL P-CHL $[unrel_i]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i}]$ NSIB-CHLSIB NP-CHL $[\neg nolc_i]$ NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[lin_i]$ P-SIB $[norc_{\leq i-1}]$ NORC-UNREL1-SIB P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i-1}]$ NSIB-CHLSIB NP-SIB $[\neg nolc_{i-1}]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\mathbf{norc}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib$ $\neg nolc$	$[lin_2]$ P-CHL $[norc_1]$ P-CHL $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib_1]$ NP-CHL $[\neg nolc_1]$ NP-CHL $[nhat]$ NHAT-UNREL-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[nsib_1]$ NP-SIB $[\neg nolc_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{norc_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_1$ $nhat$ $\neg nolc_1$ $[\neg norc]$ NOT-NORC-NHAT $[\neg nolc]$ NOT-NOLC-NHAT $[nsib]$ NSIB-NHAT	$[lin_3]$ P-CHL $[norc_2]$ P-CHL $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib_2]$ NP-CHL $[\neg nolc_2]$ NP-CHL $[nhat_{\leq 1}]$ NHAT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_2]$ P-SIB $[norc_1]$ NOLC-FLS $[ntree]$ NTREE-SINK $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[nsib_1]$ NP-SIB $[\neg nolc_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{norc_{i-1}}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_{i-1}$ $nhat_{\leq i-2}$ $\neg nolc_{i-1}$ $[\neg norc_{\leq i-2}]$ NOT-NORC-NHAT $[\neg nolc_{\leq i-2}]$ NOT-NOLC-NHAT $[nsib_{\leq i-2}]$ NSIB-NHAT	$[lin_{i+1}]$ P-CHL $[norc_i]$ P-CHL $[nodup]$ NODUP-CHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib_i]$ NP-CHL $[\neg nolc_i]$ NP-CHL $[nhat_{\leq i-1}]$ NHAT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_i]$ P-SIB $[norc_{i-1}]$ NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[nsib_{i-1}]$ NP-SIB $[\neg nolc_{i-1}]$ NP-SIB $[nhat_{\leq i-2}]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{no2d_1}$ $nsib$ $[unrel]$ UNREL-NO2D $[lin_1]$ LIN-NO2D $[nolc_{\geq 0}]$ NOLC-NO2D $[norc_{\geq 0}]$ NORC-NO2D $[\neg no2d]$ NOT-NO2D-NSIB	$[no2d_2]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[no2d_1]$ P-SIB $[nsib]$ NSIB-CHLSIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{no2d_2}$ $nsib_{\leq 1}$ $[unrel]$ UNREL-NO2D $[lin_2]$ LIN-NO2D $[nolc_{\geq 0}]$ NOLC-NO2D $[norc_{\geq 0}]$ NORC-NO2D $[\neg no2d]$ NOT-NO2D-NSIB	$[no2d_3]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[nsib_{\leq 2}]$ NSIB-CHLSIB NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[no2d_2]$ P-SIB $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{no2d_i}$ $(i > 2)$ $nsib_{\leq i-1}$ $[unrel]$ UNREL-NO2D $[lin_i]$ LIN-NO2D $[nolc_{\geq 0}]$ NOLC-NO2D $[norc_{\geq 0}]$ NORC-NO2D $[\neg no2d]$ NOT-NO2D-NSIB	$[no2d_{i+1}]$ P-CHL $[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-CHL $[nsib_{\leq i}]$ NSIB-CHLSIB NP-CHL	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[ord]$ ORD-UNREL-NODUP-DOWN $[nodup]$ NODUP-UNREL-DOWN $[ntree]$ NTREE-SINK	$[no2d_i]$ P-SIB $[nsib_{\leq i-1}]$ NSIB-CHLSIB NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page					
	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_1}$ $\mathbf{nolc}$ $\mathbf{norc}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib$	$[lin_2]$ P-CHL $[nolc_1]$ P-CHL $[norc_1]$ P-CHL $[nodup]$ NODUP-CHL $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}$ NOT-NO2D-STEP $[nsib_1]$ NP-CHL $[nhat]$ NHAT-UNREL-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_1]$ P-SIB $[nolc]$ NOLC-FLS $[unrel]$ UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-CHLSIB $[\neg norc]$ NOT-NORC-UNREL-FLS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{nolc_1}$ $\mathbf{norc_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_1$ $nhat$ $[\neg nolc]$ NOT-NOLC-NHAT $[\neg norc]$ NOT-NORC-NHAT $[nsib]$ NSIB-NHAT	$[lin_3]$ P-CHL $[nolc_2]$ P-CHL $[norc_2]$ P-CHL $[nodup]$ NODUP-CHL $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}$ NOT-NO2D-STEP $[nsib_2]$ NP-CHL $[nhat_{\leq 1}]$ NHAT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_2]$ P-SIB $[nolc_1]$ P-SIB $[norc_1]$ P-SIB $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{nolc_{i-1}}$ $\mathbf{norc_{i-1}}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_{i-1}$ $nhat_{\leq i-2}$ $[\neg nolc_{\leq i-2}]$ NOT-NOLC-NHAT $[\neg norc_{\leq i-2}]$ NOT-NORC-NHAT $[nsib_{\leq i-2}]$ NSIB-NHAT	$[lin_{i+1}]$ P-CHL $[nolc_i]$ P-CHL $[norc_i]$ P-CHL $[nodup]$ NODUP-CHL $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}$ NOT-NO2D-STEP $[nsib_i]$ NP-CHL $[nhat_{\leq i-1}]$ NHAT-UNREL-CHL NP-CHL $[\neg ord]$ NOT-ORD-UNREL-DOWN	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-UNREL-DOWN $[\neg nodup]$ NOT-NODUP-UNREL-DSC	$[lin_i]$ P-SIB $[nolc_{i-1}]$ P-SIB $[norc_{i-1}]$ P-SIB $[\neg unrel]$ NOT-UNREL-NOLC-FLS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{i-1}]$ NP-SIB $[nhat_{\leq i-2}]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
no2d $[lin]$ LIN-NO2D $[lin_1]$ LIN-UP $[no2d_1]$ NO2D-UP	$[no2d]$ P-PRN $[ord]$ ORD-NO2D-STEP $[nodup]$ NODUP-NO2D-STEP	$[lin]$ LIN-AOS $[ord]$ ORD-NO2D-STEP $[nodup]$ NODUP-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC	$[lin]$ LIN-ANC $[ord]$ ORD-NO2D-STEP $[nodup]$ NODUP-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC	$[no2d_1]$ P-SIB $[ord]$ ORD-NO2D-STEP $[nodup]$ NODUP-NO2D-STEP $[nsib]$ NSIB-CHLSIB	$[ord]$ ORD-NO2D-STEP $[nodup]$ NODUP-NO2D-STEP $[ntree]$ NTREE-SINK
(s) $ntree$ $[nhat]$ NHAT-NTREE $[nsib]$ NSIB-NHAT $[\neg norc]$ NOT-NORC-NHAT $[\neg unrel]$ NOT-UNREL-NTREE $[\neg no2d]$ NOT-NO2D-NSIB	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[ntree]$ NTREE-STEP $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
lin $\neg unrel$ $\neg no2d_{\geq 0}$ $[lin_1]$ LIN-UP $[nolc]$ NOLC-LIN $[norc]$ NORC-LIN	$[lin]$ P-PRN $[ord]$ ORD-NORC-PRN $[nodup]$ NODUP-LIN-PRNSIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL	$[lin]$ LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ P-SIB $[norc]$ NORC-PRS $[unrel]$ UNREL-NORC-PRS $[ord]$ ORD-LIN-PRS $[nodup]$ NODUP-LIN-PRNSIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-CHLSIB $[\neg nolc]$ NOT-NOLC-UNREL-PRS	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
lin$_1$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nhat$ $[lin_2]$ LIN-UP $[\neg nolc]$ NOT-NOLC-NHAT $[\neg norc]$ NOT-NORC-NHAT $[nsib]$ NSIB-NHAT	$[lin]$ P-PRN $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_1]$ LIN-AOS $[\neg unrel]$ NOT-UNREL-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nhat]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg unrel]$ NOT-UNREL-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
$\mathbf{lin_i}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nhat_{\leq i-1}$ $[lin_{i+1}]$ $[\neg nolc_{\leq i-1}]$ $[\neg norc_{\leq i-1}]$ $[nsib_{\leq i-1}]$	$(i > 1)$ LIN-UP NOT-NOLC-NHAT NOT-NORC-NHAT NSIB-NHAT	$[lin_{i-1}]$ $[\neg unrel]$ $[\neg no2d_{\geq 0}]$ $[nhat_{\leq i-2}]$ $[\neg ord]$ $[\neg nodup]$	P-PRN NOT-UNREL-PRNCHL NOT-NO2D-STEP NP-PRNANC NOT-ORD-NORC-PRN NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ $[\neg unrel]$ $[\neg no2d_{\geq 0}]$ $[nhat_{\leq i-1}]$ $[\neg ord]$ $[\neg nodup]$	LIN-AOS NOT-UNREL-ANC NOT-NO2D-STEP NP-AOS NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ $[\neg unrel]$ $[\neg no2d_{\geq 0}]$ $[nhat_{\leq i-2}]$ $[\neg ord]$ $[\neg nodup]$	LIN-ANC NOT-UNREL-ANC NOT-NO2D-STEP NP-PRNANC NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_i]$ $[\neg unrel]$ $[\neg no2d_{\geq 0}]$ $[nhat_{\leq i-1}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB NOT-UNREL-NORC-PRS NOT-NO2D-STEP NHAT-SIB NP-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\mathbf{nolc}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib$ $\neg norc$		$[lin]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$	P-PRN NOT-NO2D-STEP NOT-UNREL-PRNCHL	$[lin_1]$ $[nolc]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib]$ $[\neg norc]$ $[\neg ord]$ $[\neg nodup]$	LIN-AOS NOLC-LIN-AOS NOT-NO2D-STEP NOT-UNREL-ANC NP-AOS NP-AOS NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[\neg ord]$ $[\neg nodup]$	LIN-ANC NOT-NO2D-STEP NOT-UNREL-ANC NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_1]$ $[\neg no2d_{\geq 0}]$ $[nhat]$ $[\neg ord]$ $[\neg nodup]$	P-SIB NOT-NO2D-STEP NHAT-NORC-PRS NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{nolc_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_1$ $\neg norc_1$ $nhat$ $[\neg nolc]$ $[\neg norc]$ $[nsib]$	 NOT-NOLC-NHAT NOT-NORC-NHAT NSIB-NHAT	$[lin_1]$ $[nolc]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib]$ $[\neg norc]$ $[\neg ord]$ $[\neg nodup]$	P-PRN P-PRN NOT-NO2D-STEP NOT-UNREL-PRNCHL NP-PRNANC NP-PRNANC NOT-ORD-NORC-PRN NOT-NODUP-NSIB-PRNSIB	$[lin_2]$ $[nolc_1]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib_1]$ $[\neg norc_1]$ $[nhat]$ $[\neg ord]$ $[\neg nodup]$	LIN-AOS NOLC-LIN-AOS NOT-NO2D-STEP NOT-UNREL-ANC NP-AOS NP-AOS NP-AOS NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_1]$ $[nolc]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib]$ $[\neg norc]$ $[\neg ord]$ $[\neg nodup]$	LIN-ANC NOLC-LIN-ANC NOT-NO2D-STEP NOT-UNREL-ANC NP-ANC NP-ANC NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_2]$ $[nolc_1]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib_1]$ $[\neg norc_1]$ $[nhat]$ $[\neg ord]$ $[\neg nodup]$	P-SIB P-SIB NOT-NO2D-STEP NOT-UNREL-NORC-PRS NP-SIB NP-SIB NHAT-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA
$\mathbf{lin_i}$ $\mathbf{nolc_{i-1}}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_{i-1}$ $\neg norc_{i-1}$ $nhat_{\leq i-2}$ $[\neg nolc_{\leq i-2}]$ $[\neg norc_{\leq i-2}]$ $[nsib_{\leq i-2}]$	$(i > 2)$ NOT-NOLC-NHAT NOT-NORC-NHAT NSIB-NHAT	$[lin_{i-1}]$ $[nolc_{i-2}]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib_{i-2}]$ $[\neg norc_{i-2}]$ $[\neg ord]$ $[\neg nodup]$	P-PRN P-PRN NOT-NO2D-STEP NOT-UNREL-PRNCHL NP-PRNANC NP-PRNANC NOT-ORD-NORC-PRN NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ $[nolc_{i-1}]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib_{i-1}]$ $[\neg norc_{i-1}]$ $[nhat_{\leq i-2}]$ $[\neg ord]$ $[\neg nodup]$	LIN-AOS NOLC-LIN-AOS NOT-NO2D-STEP NOT-UNREL-ANC NP-AOS NP-AOS NP-AOS NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ $[nolc_{i-2}]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib_{i-2}]$ $[\neg norc_{i-2}]$ $[nhat_{\leq i-3}]$ $[\neg ord]$ $[\neg nodup]$	LIN-ANC NOLC-LIN-ANC NOT-NO2D-STEP NOT-UNREL-ANC NP-ANC NP-ANC NP-PRNANC NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA	$[lin_i]$ $[nolc_{i-1}]$ $[\neg no2d_{\geq 0}]$ $[\neg unrel]$ $[nsib_{i-1}]$ $[\neg norc_{i-1}]$ $[nhat_{\leq i-2}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB P-SIB NOT-NO2D-STEP NOT-UNREL-NORC-PRS NP-SIB NP-SIB NHAT-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[ntree]$ $[\neg ord]$ $[\neg nodup]$	NTREE-SINK NOT-ORD-NO2D-FPA NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_1}$ $\mathbf{nolc}$ $\mathbf{unrel}$ $\neg no2d_{\geq 0}$ $nsib$ $\neg norc$	$[lin]$ P-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_1]$ LIN-AOS $[nolc]$ NOLC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-AOS $[\neg norc]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nhat]$ NHAT-NORC-PRS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{nolc_1}$ $\mathbf{norc}$ $\mathbf{unrel_1}$ $\neg no2d_{\geq 0}$ $nsib_{\leq 1}$ $\neg norc_1$ $[unrel]$ UNREL-DOWN	$[lin_1]$ P-PRN $[nolc]$ P-PRN $[unrel]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NP-PRNANC $[\neg norc]$ NP-PRNANC $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_2]$ LIN-AOS $[nolc_1]$ NOLC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_1]$ NP-AOS $[\neg norc_1]$ NP-AOS $[nhat]$ NHAT-NSIB-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ LIN-ANC $[nolc]$ NOLC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-PRNANC $[\neg norc]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_2]$ P-SIB $[nolc_1]$ P-SIB $[norc]$ NORC-PRS $[unrel_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-SIB $[\neg norc_1]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{nolc_{i-1}}$ $\mathbf{norc_{\leq i-2}}$ $\mathbf{unrel_{i-1}}$ $\neg no2d_{\geq 0}$ $nsib_{\leq i-1}$ $\neg norc_{i-1}$ $[unrel_{\leq i-1}]$ UNREL-DOWN	$[lin_{i-1}]$ P-PRN $[nolc_{i-2}]$ P-PRN $[norc_{\leq i-3}]$ P-PRN $[unrel_{i-2}]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i-2}]$ NP-PRNANC $[\neg norc_{i-2}]$ NP-PRNANC $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ LIN-AOS $[nolc_{i-1}]$ NOLC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{\leq i-1}]$ NP-AOS $[\neg norc_{i-1}]$ NP-AOS $[nhat_{\leq i-2}]$ NHAT-NSIB-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ LIN-ANC $[nolc_{i-2}]$ NOLC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{\leq i-2}]$ NP-PRNANC $[\neg norc_{i-2}]$ NP-PRNANC $[nhat_{\leq i-3}]$ NHAT-NSIB-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_i]$ P-SIB $[nolc_{i-1}]$ P-SIB $[norc_{\leq -i2}]$ NORC-PRS P-SIB $[unrel_{i-1}]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i-1}]$ NSIB-CHLSIB NP-SIB $[\neg norc_{i-1}]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_1}$ $\mathbf{norc}$ $\mathbf{unrel}$ $\neg no2d_{\geq 0}$ $nsib$ $\neg nolc$	$[lin]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg unrel]$ NOT-UNREL-PRNCHL $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_1]$ LIN-AOS $[norc]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-AOS $[\neg nolc]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ P-SIB $[norc]$ NORC-PRS $[unrel]$ UNREL-NORC-PRS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-CHLSIB $[\neg nolc]$ NOT-NOLC-PRS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{norc_{\leq 1}}$ $\mathbf{unrel_1}$ $\neg no2d_{\geq 0}$ $nsib_{\leq 1}$ $\neg nolc_1$	$[lin_1]$ P-PRN $[norc]$ P-PRN $[unrel]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NP-PRNANC $[\neg nolc]$ NP-PRNANC $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_2]$ LIN-AOS $[norc_1]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{\leq 1}]$ NP-AOS $[\neg nolc_1]$ NP-AOS $[nhat]$ NHAT-NSIB-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ LIN-ANC $[norc]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-PRNANC $[\neg nolc]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_2]$ P-SIB $[norc_1]$ NORC-PRS P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-SIB $[\neg nolc_1]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{norc_{\leq i-1}}$ $\mathbf{unrel_{i-1}}$ $\neg no2d_{\geq 0}$ $nsib_{\leq i-1}$ $\neg nolc_{i-1}$ $[unrel_{\leq i-1}]$ UNREL-DOWN	$[lin_{i-1}]$ P-PRN $[norc_{\leq i-2}]$ P-PRN $[unrel_{i-2}]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i-2}]$ NP-PRNANC $[\neg nolc_{i-2}]$ NP-PRNANC $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ LIN-AOS $[norc_{i-1}]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{\leq i-1}]$ NP-AOS $[\neg nolc_{i-1}]$ NP-AOS $[nhat_{\leq i-2}]$ NHAT-NSIB-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ LIN-ANC $[norc_{i-2}]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{\leq i-2}]$ NP-PRNANC $[\neg nolc_{i-2}]$ NP-PRNANC $[nhat_{\leq i-3}]$ NHAT-NSIB-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_i]$ P-SIB $[norc_{i-1}]$ NORC-PRS P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib_{\leq i-1}]$ NSIB-CHLSIB NP-SIB $[\neg nolc_{i-1}]$ NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\mathbf{norc}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib$ $\neg nolc$	$[lin]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL	$[lin_1]$ LIN-AOS $[norc]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-AOS $[\neg nolc]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC	$[lin_1]$ P-SIB $[norc]$ NORC-PRS $[unrel]$ UNREL-NORC-PRS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-CHLSIB $[\neg nolc]$ NOT-NOLC-PRS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_2}$ $\mathbf{norc_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_1$ $nhat$ $\neg nolc_1$ $[\neg norc]$ NOT-NORC-NHAT $[\neg nolc]$ NOT-NOLC-NHAT $[nsib]$ NSIB-NHAT	$[lin_1]$ P-PRN $[norc]$ P-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib]$ NP-PRNANC $[\neg nolc]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_2]$ LIN-AOS $[norc_1]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_1]$ NP-AOS $[\neg nolc_1]$ NP-AOS $[nhat]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ LIN-ANC $[norc]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-PRNANC $[\neg nolc]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_2]$ P-SIB $[norc_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nsib_1]$ NP-SIB $[\neg nolc_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{norc_{i-1}}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_{i-1}$ $nhat_{\leq i-2}$ $\neg nolc_{i-1}$ $[\neg norc_{\leq i-2}]$ NOT-NORC-NHAT $[\neg nolc_{\leq i-2}]$ NOT-NOLC-NHAT $[nsib_{\leq i-2}]$ NSIB-NHAT	$[lin_{i-1}]$ P-PRN $[norc_{i-2}]$ P-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib_{i-2}]$ NP-PRNANC $[\neg nolc_{i-2}]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ LIN-AOS $[norc_{i-1}]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{i-1}]$ NP-AOS $[\neg nolc_{i-1}]$ NP-AOS $[nhat_{\leq i-2}]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ LIN-ANC $[norc_{i-2}]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{i-2}]$ NP-PRNANC $[\neg nolc_{i-2}]$ NP-PRNANC $[nhat_{\leq i-3}]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_i]$ P-SIB $[norc_{i-1}]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nsib_{i-1}]$ NP-SIB $[\neg nolc_{i-1}]$ NP-SIB $[nhat_{\leq i-2}]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{no2d_1}$ $nsib$ $[unrel]$ UNREL-NO2D $[lin_1]$ LIN-NO2D $[nolc_{\geq 0}]$ NOLC-NO2D $[norc_{\geq 0}]$ NORC-NO2D $[\neg no2d]$ NOT-NO2D-NSIB	$[no2d]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_1]$ LIN-AOS $[nolc]$ NOLC-LIN-AOS $[norc]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[no2d_1]$ P-SIB $[nsib]$ NSIB-CHLSIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{no2d_2}$ $nsib_{\leq 1}$ $[unrel]$ UNREL-NO2D $[lin_2]$ LIN-NO2D $[nolc_{\geq 0}]$ NOLC-NO2D $[norc_{\geq 0}]$ NORC-NO2D $[\neg no2d]$ NOT-NO2D-NSIB	$[no2d_1]$ P-PRN $[ord]$ ORD-NORC-PRN $[nsib]$ NP-PRNANC $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_2]$ LIN-AOS $[nolc_1]$ NOLC-LIN-AOS $[norc_1]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_1]$ NP-AOS $[nhat]$ NHAT-NSIB-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ LIN-ANC $[nolc]$ NOLC-LIN-ANC $[norc]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[no2d_2]$ P-SIB $[nsib_{\leq 1}]$ NSIB-CHLSIB NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{no2d_i}$ $(i > 2)$ $nsib_{\leq i-1}$ $[unrel]$ UNREL-NO2D $[lin_i]$ LIN-NO2D $[nolc_{\geq 0}]$ NOLC-NO2D $[norc_{\geq 0}]$ NORC-NO2D $[\neg no2d]$ NOT-NO2D-NSIB	$[no2d_{i-1}]$ P-PRN $[ord]$ ORD-NORC-PRN $[nsib_{\leq i-2}]$ NP-PRNANC $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ LIN-AOS $[nolc_{i-1}]$ NOLC-LIN-AOS $[norc_{i-1}]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{i-1}]$ NP-AOS $[nhat_{\leq i-2}]$ NHAT-NSIB-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ LIN-ANC $[nolc_{i-2}]$ NOLC-LIN-ANC $[norc_{i-2}]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{i-2}]$ NP-ANC $[nhat_{\leq i-3}]$ NHAT-NSIB-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[no2d_i]$ P-SIB $[nsib_{\leq i-1}]$ NSIB-CHLSIB NP-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_1}$ $\mathbf{nolc}$ $\mathbf{norc}$	$[lin]$ P-PRN $[ord]$ ORD-NORC-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL	$[lin_1]$ LIN-AOS $[nolc]$ NOLC-LIN-AOS $[norc]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin]$ LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ P-SIB $[norc]$ NORC-PRS $[unrel]$ UNREL-NORC-PRS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[nsib]$ NSIB-CHLSIB $[\neg nolc]$ NOT-NOLC-UNREL-PRS $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{lin_2}$ $\mathbf{nolc_1}$ $\mathbf{norc_1}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_1$ $nhat$ $[\neg nolc]$ NOT-NOLC-NHAT $[\neg norc]$ NOT-NORC-NHAT $[nsib]$ NSIB-NHAT	$[lin_1]$ P-PRN $[nolc]$ P-PRN $[norc]$ P-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_2]$ LIN-AOS $[nolc_1]$ NOLC-LIN-AOS $[norc_1]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib_1]$ NP-AOS $[nhat]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_1]$ LIN-ANC $[nolc]$ NOLC-LIN-ANC $[norc]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-ANC $[\neg unrel]$ NOT-UNREL-ANC $[nsib]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_2]$ P-SIB $[nolc_1]$ P-SIB $[norc_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nsib_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{lin_i}$ $(i > 2)$ $\mathbf{nolc_{i-1}}$ $\mathbf{norc_{i-1}}$ $\neg unrel$ $\neg no2d_{\geq 0}$ $nsib_{i-1}$ $nhat_{\leq i-2}$ $[\neg nolc_{\leq i-2}]$ NOT-NOLC-NHAT $[\neg norc_{\leq i-2}]$ NOT-NORC-NHAT $[nsib_{\leq i-2}]$ NSIB-NHAT	$[lin_{i-1}]$ P-PRN $[nolc_{i-2}]$ P-PRN $[norc_{i-2}]$ P-PRN $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-PRNCHL $[nsib_{i-2}]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NORC-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[lin_i]$ LIN-AOS $[nolc_{i-1}]$ NOLC-LIN-AOS $[norc_{i-1}]$ NORC-LIN-AOS $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{i-1}]$ NP-AOS $[nhat_{\leq i-2}]$ NP-AOS $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_{i-1}]$ LIN-ANC $[nolc_{i-2}]$ NOLC-LIN-ANC $[norc_{i-2}]$ NORC-LIN-ANC $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC $[nsib_{i-2}]$ NP-PRNANC $[nhat_{\leq i-3}]$ NP-PRNANC $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_i]$ P-SIB $[nolc_{i-1}]$ P-SIB $[norc_{i-1}]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nsib_{i-1}]$ NP-SIB $[nhat_{\leq i-2}]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[ntree]$ NTREE-SINK $[\neg ord]$ NOT-ORD-NO2D-FPA $[\neg nodup]$ NOT-NODUP-NO2D-FPA

Appendix B

Correctness of the Automata for Sloppy Evaluation Plans

B.1 Correctness of the A_{ord}^{sloppy} Automaton

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
ord **nodup** **no2d** $[lin]$ $[lin_1]$ $[no2d_1]$ $[gen]$	LIN-NO2D LIN-UP NO2D-UP GEN-NO2D	$[ord]$ $[nodup]$ $[gen]$ $[nsib]$	ORD-NO2D-STEP NODUP-NO2D-STEP P-CHL NSIB-CHLSIB	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-NO2D-STEP NTREE-SINK NOT-ORD-NTREE-GT	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-NO2D-STEP NTREE-SINK NOT-ORD-NTREE-GT	$[ord]$ $[nodup]$ $[gen]$ $[nsib]$	ORD-NO2D-STEP NODUP-NO2D-STEP GEN-STEP NSIB-CHLSIB	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-NO2D-STEP NTREE-SINK NOT-ORD-NTREE-GT
ord **nodup** **gen** $nsib$ $[unrel]$ $[ord_1]$ $[\neg no2d]$	UNREL-GEN ORD-GEN NOT-NO2D-NSIB	$[gen]$ $[ord]$ $[nodup]$ $[nsib]$	GEN-STEP ORD-UNREL-NODUP-DOWN NODUP-UNREL-DOWN NSIB-CHLSIB	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord_1]$ $[gen]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB GEN-STEP NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
ord $ntree$ $\neg ord_{\geq 1}$ $[\neg unrel]$ $[nhat]$ $[nsib]$ $[\neg no2d]$	NOT-UNREL-NTREE NHAT-NTREE NSIB-NHAT NOT-NO2D-NSIB	$[ord_1]$ $[ntree]$ $[\neg ord]$ $[\neg ord_{\geq 2}]$	ORD-CHL NTREE-STEP NOT-ORD-UNREL-DOWN NOT-ORD-NTREE-GT	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-UNREL-FPD NOT-NODUP-UNREL-DSC	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-UNREL-FPD NOT-NODUP-UNREL-DSC	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NSIB-SIB NOT-ORD-FPSIB NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
$\mathbf{ord_1}$ $\neg ord$ $\neg ord_{\geq 2}$ $ntree$ $[\neg unrel]$ $[nhat]$ $[nsib]$ $[\neg no2d]$	NOT-UNREL-NTREE NHAT-NTREE NSIB-NHAT NOT-NO2D-NSIB	$[ord_2]$ $[ntree]$ $[\neg ord_{\leq 1}]$ $[\neg ord_{\geq 3}]$	ORD-CHL NTREE-STEP NOT-ORD-NTREE-LT NOT-ORD-NTREE-GT	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-UNREL-FPD NOT-NODUP-UNREL-DSC	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-UNREL-FPD NOT-NODUP-UNREL-DSC	$[ord_1]$ $[ntree]$ $[\neg ord]$ $[\neg ord_{\geq 2}]$	ORD-SIB NTREE-STEP NOT-ORD-NTREE-LT NOT-ORD-NTREE-GT	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
$\mathbf{ord_i}$ $\neg ord_{\leq i-1}$ $\neg ord_{\geq i+1}$ $ntree$ $[\neg unrel]$ $[nhat]$ $[nsib]$ $[\neg no2d]$	$(i > 1)$ NOT-UNREL-NTREE NHAT-NTREE NSIB-NHAT NOT-NO2D-NSIB	$[ord_{i+1}]$ $[ntree]$ $[\neg ord_{\leq i}]$ $[\neg ord_{\geq i+2}]$	ORD-CHL NTREE-STEP NOT-ORD-NTREE-LT NOT-ORD-NTREE-GT	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-UNREL-FPD NOT-NODUP-UNREL-DSC	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-UNREL-FPD NOT-NODUP-UNREL-DSC	$[ord_i]$ $[ntree]$ $[\neg ord_{\leq i-1}]$ $[\neg ord_{\geq i+1}]$	ORD-SIB NTREE-STEP NOT-ORD-NTREE-LT NOT-ORD-NTREE-GT	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
ord **gen** $\neg nodup$ $[ord_1]$	ORD-GEN	$[gen]$ $[ord_1]$ $[\neg ord]$ $[\neg nodup]$	GEN-STEP ORD-CHL NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[gen]$ $[ord_1]$ $[\neg ord]$ $[\neg nodup]$	GEN-STEP ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP

Continued on next page

Continued from previous page

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
$\mathbf{ord_1}$ $\mathbf{gen}$ $\neg ord$ $\neg nodup$		$[gen]$ $[ord_2]$ $[\neg ord_{\leq 1}]$ $[\neg nodup]$	GEN-STEP ORD-CHL NOT-ORD-DOWN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[gen]$ $[ord_1]$ $[\neg ord]$ $[\neg nodup]$	GEN-STEP ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_i}$ $\mathbf{gen}$ $\neg ord_{\leq i-1}$ $\neg nodup$	$(i > 1)$	$[gen]$ $[ord_{i+1}]$ $[\neg ord_{\leq i}]$ $[\neg nodup]$	GEN-STEP ORD-CHL NOT-ORD-DOWN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[gen]$ $[ord_i]$ $[\neg ord_{\leq i-1}]$ $[\neg nodup]$	GEN-STEP ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_{\geq 0}}$ $\mathbf{lin}$ $\mathbf{nodup}$ $\neg unrel$ $[lin_1]$ $[norc]$ $[\neg no2d]$	LIN-UP NORC-LIN NOT-NO2D-UNREL	$[lin_1]$ $[nodup]$ $[ord_{\geq 1}]$ $[\neg ord]$ $[nsib]$	P-CHL NODUP-CHL ORD-CHL NOT-ORD-UNREL-DOWN NSIB-CHLSIB	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-UNREL-FPD NTREE-SINK	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-UNREL-FPD NTREE-SINK	$[lin_1]$ $[nodup]$ $[ord_{\geq 1}]$ $[\neg ord]$ $[nsib]$	P-SIB NODUP-LIN-PRNSIB ORD-SIB NOT-ORD-UNREL-FLS NSIB-CHLSIB	$[\neg ord_{ge0}]$ $[ntree]$	NOT-ORD-UNREL-FPD NTREE-SINK
$\mathbf{ord_{\geq 1}}$ $\mathbf{lin_1}$ $\mathbf{nodup}$ $\neg ord$ $nsib$ $[\neg no2d]$	NOT-NO2D-NSIB	$[lin_2]$ $[nodup]$ $[ord_2]$ $[\neg ord_{\leq 1}]$ $[nsib_{\leq 1}]$	P-CHL NODUP-CHL ORD-CHL NOT-ORD-DOWN NSIB-CHLSIB NP-CHL	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-DOWN NOT-ORD-NO2D-REC NTREE-SINK	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-DOWN NOT-ORD-NO2D-REC NTREE-SINK	$[lin_i]$ $[ord_{\geq i}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-SIB	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NTREE-SINK
$\mathbf{ord_{\geq i}}$ $\mathbf{lin_i}$ $\mathbf{nodup}$ $\neg ord_{\leq i-1}$ $nsib_{\leq i-1}$ $[\neg no2d]$	$(i > 1)$ NOT-NO2D-NSIB	$[lin_{i+1}]$ $[nodup]$ $[ord_{\geq i+1}]$ $[\neg ord_{\leq i}]$ $[nsib_{\leq i}]$	P-CHL NODUP-CHL ORD-CHL NOT-ORD-DOWN NSIB-CHLSIB NP-CHL	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-DOWN NOT-ORD-NO2D-REC NTREE-SINK	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-DOWN NOT-ORD-NO2D-REC NTREE-SINK	$[lin_i]$ $[ord_{\geq i}]$ $[\neg ord_{\leq i-1}]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NSIB-SIB NOT-ORD-FPSIB NOT-NODUP-NSIB-SIB	$[\neg ord_{\geq 0}]$ $[ntree]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NTREE-SINK
$\mathbf{ord_{\geq 0}}$ $\mathbf{lin}$ $\neg nodup$ $[lin_1]$	LIN-UP	$[lin_1]$ $[ord_{\geq 1}]$ $[\neg ord]$ $[\neg nodup]$	P-CHL ORD-CHL NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[lin_1]$ $[ord_{\geq 1}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_{\geq 1}}$ $\mathbf{lin_1}$ $\neg ord$ $\neg nodup$		$[lin_2]$ $[ord_{\geq 2}]$ $[\neg ord_{\leq 1}]$ $[\neg nodup]$	P-CHL ORD-CHL NOT-ORD-NODUP-CHLSIB NP-CHL NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[lin_i]$ $[ord_{\geq i}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP

Continued on next page

Continued from previous page

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
$\mathbf{ord}_{\geq \mathbf{i}}$ $\mathbf{lin_i}$ $\neg ord_{\leq i-1}$ $\neg nodup$	$(i > 1)$	$[lin_{i+1}]$ $[ord_{\geq i+1}]$ $[\neg ord_{\leq i}]$ $[\neg nodup]$	P-CHL ORD-CHL NOT-ORD-NODUP-CHLSIB NP-CHL NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[lin_i]$ $[ord_{\geq i}]$ $[\neg ord_{\leq i-1}]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord}_{\geq \mathbf{0}}$ $\mathbf{lin_1}$ $\mathbf{unrel}$ $\mathbf{nodup}$ $nsib$ $[\neg no2d]$	NOT-NO2D-NSIB	$[ord_{\geq 0}]$ $[lin_2]$ $[unrel_1]$ $[nodup]$ $[nsib]$	ORD-UNREL-NODUP-DOWN ORD-CHL P-CHL P-CHL NODUP-CHL NSIB-CHLSIB	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord_{\geq 1}]$ $[lin_1]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ $[\neg ord_{\geq 0}]$	NOT-NODUP-NO2D-FPA NOT-ORD-NSIB-FPA
$\mathbf{ord}_{\geq \mathbf{0}}$ $\mathbf{lin_2}$ $\mathbf{unrel_1}$ $\mathbf{nodup}$ $nsib$ $[unrel]$ $[\neg no2d]$	UNREL-DOWN NOT-NO2D-NSIB	$[ord_{\geq 0}]$ $[lin_3]$ $[unrel_2]$ $[nodup]$ $[nsib]$	ORD-UNREL-NODUP-DOWN ORD-CHL P-CHL P-CHL NODUP-CHL NSIB-CHLSIB	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord_{\geq 1}]$ $[lin_2]$ $[unrel_1]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ $[\neg ord_{\geq 0}]$	NOT-NODUP-NO2D-FPA NOT-ORD-NSIB-FPA
$\mathbf{ord}_{\geq \mathbf{0}}$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\mathbf{nodup}$ $nsib$ $[unrel]$ $[\neg no2d]$	$(i > 2)$ UNREL-DOWN NOT-NO2D-NSIB	$[ord_{\geq 0}]$ $[lin_{i+1}]$ $[unrel_i]$ $[nodup]$ $[nsib]$	ORD-UNREL-NODUP-DOWN ORD-CHL P-CHL P-CHL NODUP-CHL NSIB-CHLSIB	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord]$ $[ntree]$ $[\neg ord_{\geq 1}]$	ORD-UNREL-NODUP-DOWN NTREE-SINK NOT-ORD-NTREE-GT	$[ord_{\geq 1}]$ $[lin_i]$ $[unrel_{i-1}]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ $[\neg ord_{\geq 0}]$	NOT-NODUP-NO2D-FPA NOT-ORD-NSIB-FPA
$\mathbf{ord}_{\geq \mathbf{0}}$ $\mathbf{lin_1}$ $\mathbf{unrel}$ $\neg nodup$		$[ord_{\geq 1}]$ $[lin_2]$ $[unrel_1]$ $[\neg ord]$ $[\neg nodup]$	ORD-CHL P-CHL P-CHL NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[ord_{\geq 1}]$ $[lin_1]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord}_{\geq \mathbf{0}}$ $\mathbf{lin_2}$ $\mathbf{unrel_1}$ $\neg nodup$ $[unrel]$	UNREL-DOWN	$[ord_{\geq 1}]$ $[lin_3]$ $[unrel_2]$ $[\neg ord]$ $[\neg nodup]$	ORD-CHL P-CHL P-CHL NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[ord_{\geq 1}]$ $[lin_2]$ $[unrel_1]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP

Continued on next page

Continued from previous page

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
$\mathbf{ord_{\geq 0}}$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\neg nodup$ $[unrel]$	$(i > 2)$ UNREL-DOWN	$[ord_{\geq 1}]$ $[lin_{i+1}]$ $[unrel_i]$ $[\neg ord]$ $[\neg nodup]$	ORD-CHL P-CHL P-CHL NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[ord_{\geq 1}]$ $[lin_i]$ $[unrel_{i-1}]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_{\geq 1}}$ $\mathbf{lin_2}$ $\mathbf{unrel_1}$ $\neg ord$ $\neg nodup$ $[unrel]$	UNREL-DOWN	$[ord_{\geq 2}]$ $[lin_2]$ $[unrel_1]$ $[\neg ord_{\leq 1}]$ $[\neg nodup]$	ORD-CHL P-CHL P-CHL NOT-ORD-DOWN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[ord_{\geq 1}]$ $[lin_2]$ $[unrel_1]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_{\geq 1}}$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\neg ord$ $\neg nodup$	$(i > 2)$	$[ord_{\geq 2}]$ $[lin_{i+1}]$ $[unrel_i]$ $[\neg ord_{\leq 1}]$ $[\neg nodup]$	ORD-CHL P-CHL P-CHL NOT-ORD-DOWN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[ord_{\geq 1}]$ $[lin_i]$ $[unrel_{i-1}]$ $[\neg ord]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_{\geq j}}$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\neg ord_{\leq j-1}$ $\neg nodup$	$(i > 2, j > 1)$	$[ord_{\geq j+1}]$ $[lin_{i+1}]$ $[unrel_i]$ $[\neg ord_{\leq j}]$ $[\neg nodup]$	ORD-CHL P-CHL P-CHL NOT-ORD-DOWN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[ord_{\geq j}]$ $[lin_i]$ $[unrel_{i-1}]$ $[\neg ord_{\leq j-1}]$ $[\neg nodup]$	ORD-SIB P-SIB P-SIB NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
(s) $\neg ord_{\geq 0}$ $\neg nodup$		$[negord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-DOWN NOT-NODUP-STEP	$[negord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[negord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[negord_{\geq 0}]$ NOT-ORD-FPSIB $[\neg nodup]$	NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[negord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
(s') $\neg ord_{\geq 0}$ $\neg ntree$ $[nhat]$ $[nsib]$	NHAT-NTREE NSIB-NHAT	$[negord_{\geq 0}]$ $[\neg ntree]$	NOT-ORD-DOWN NTREE-STEP	$[negord_{\geq 0}]$ $[\neg ntree]$	NOT-ORD-DOWN NTREE-STEP	$[negord_{\geq 0}]$ $[\neg ntree]$	NOT-ORD-DOWN NTREE-STEP	$[negord_{\geq 0}]$ $[\neg ntree]$	NOT-ORD-NSIB-SIB NOT-ORD-FPSIB NTREE-STEP	$[negord_{\geq 0}]$ $[\neg ntree]$	NOT-ORD-NSIB-FPA NTREE-STEP

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
ord **nodup** **no2d** [lin] [lin_1] [$no2d_1$] [gen]	LIN-NO2D LIN-UP NO2D-UP GEN-NO2D	[$no2d$] [ord] [$nodup$]	P-PRN ORD-NO2D-STEP NODUP-NO2D-STEP	[lin] [$ord_{\geq 0}$] [$nodup$] [$\neg unrel$]	LIN-AOS ORD-NO2D-STEP ORD-LIN NODUP-NO2D-STEP NOT-UNREL-ANC	[lin] [$ord_{\geq 0}$] [$nodup$] [$\neg unrel$]	LIN-ANC ORD-NO2D-STEP ORD-LIN NODUP-NO2D-STEP NOT-UNREL-ANC	[gen] [ord] [$nodup$] [$nsib$]	GEN-STEP ORD-NO2D-STEP NODUP-NO2D-STEP NSIB-CHLSIB	[ord] [$ntree$] [$\neg ord_{\geq 1}$]	ORD-NO2D-STEP NTREE-SINK NOT-ORD-NTREE-GT
ord **nodup** **gen** $nsib$ [$unrel$] [ord_1] [$\neg no2d$]	UNREL-GEN ORD-GEN NOT-NO2D-NSIB	[gen] [ord] [$\neg nodup$]	GEN-STEP ORD-PRN NOT-NODUP-NSIB-PRNSIB	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[ord_1] [gen] [$\neg nodup$] [$\neg ord$]	ORD-SIB GEN-STEP NOT-NODUP-NSIB-PRNSIB NOT-ORD-NSIB-SIB	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
ord $ntree$ $\neg ord_{\geq 1}$ [$\neg unrel$] [$nhat$] [$nsib$] [$\neg no2d$]	NOT-UNREL-NTREE NHAT-NTREE NSIB-NHAT NOT-NO2D-NSIB	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-UP NOT-NODUP-NSIB-PRNSIB	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-SIB NOT-ORD-FPSIB NOT-NODUP-NSIB-PRNSIN	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
ord$_1$ $\neg ord$ $\neg ord_{\geq 2}$ $ntree$ [$\neg unrel$] [$nhat$] [$nsib$] [$\neg no2d$]	NOT-UNREL-NTREE NHAT-NTREE NSIB-NHAT NOT-NO2D-NSIB	[ord] [$ntree$] [$\neg ord_{\geq 1}$]	ORD-PRN NTREE-STEP NOT-ORD-NTREE-GT	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[ord_1] [$ntree$] [$\neg ord_{\geq 2}$]	ORD-SIB NTREE-STEP NOT-ORD-FPSIB	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA
ord$_1$ $\neg ord_{\leq i-1}$ $\neg ord_{\geq i+1}$ $ntree$ [$\neg unrel$] [$nhat$] [$nsib$] [$\neg no2d$]	($i > 1$) NOT-UNREL-NTREE NHAT-NTREE NSIB-NHAT NOT-NO2D-NSIB	[ord_{i-1}] [$ntree$] [$\neg ord_{\leq i-2}$] [$\neg ord_{\geq i}$]	ORD-PRN NTREE-STEP NOT-ORD-NTREE-GT NOT-ORD-NTREE-GT	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA	[ord_i] [$ntree$] [$\neg ord_{\leq i-1}$] [$\neg ord_{\geq i+1}$]	ORD-SIB NTREE-STEP NOT-ORD-NSIB-SIB NOT-ORD-FPSIB NOT-ORD-FPSIB	[$\neg ord_{\geq 0}$] [$\neg nodup$]	NOT-ORD-NSIB-FPA NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
$\mathbf{ord}$ $\mathbf{gen}$ $\neg nodup$ $[ord_1]$	ORD-GEN	$[gen]$ $[ord]$ $[\neg nodup]$	GEN-STEP ORD-PRN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[gen]$ $[ord_1]$ $[\neg ord]$ $[\neg nodup]$	GEN-STEP ORD-SIB NOT-ORD-NSIB-SIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_1}$ $\mathbf{gen}$ $\neg ord$ $\neg nodup$		$[gen]$ $[ord]$ $[\neg nodup]$	GEN-STEP ORD-PRN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[gen]$ $[ord_1]$ $[\neg ord]$ $[\neg nodup]$	GEN-STEP ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_i}$ $\mathbf{gen}$ $\neg ord_{\leq i-1}$ $\neg nodup$	$(i > 1)$	$[gen]$ $[ord_{i-1}]$ $[\neg ord_{\leq i-2}]$ $[\neg nodup]$	GEN-STEP ORD-PRN NOT-ORD-UP NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[gen]$ $[ord_i]$ $[\neg ord_{\leq i-1}]$ $[\neg nodup]$	GEN-STEP ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP
$\mathbf{ord_{\geq 0}}$ $\mathbf{lin}$ $\mathbf{nodup}$ $\neg unrel$ $[lin_1]$ $[norc]$ $[\neg no2d]$	LIN-UP NORC-LIN NOT-NO2D-UNREL	$[lin]$ $[nodup]$ $[ord_{\geq 0}]$ $[\neg unrel]$	P-PRN NODUP-LIN-PRNSIB ORD-PRN NOT-UNREL-PRNCHL	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA	$[lin_1]$ $[unrel]$ $[nodup]$ $[ord_{\geq 0}]$ $[nsib]$	P-SIB NORC-PRS NODUP-LIN-PRNSIB ORD-LIN-PRS ORD-SIB NSIB-CHLSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA
$\mathbf{ord_{\geq 1}}$ $\mathbf{lin_1}$ $\mathbf{nodup}$ $\neg ord$ $nsib$ $[\neg no2d]$	NOT-NO2D-NSIB	$[lin]$ $[ord_{\geq 0}]$ $[\neg nodup]$	P-PRN ORD-PRN NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA	$[lin_1]$ $[ord_{\geq 1}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NSIB-SIB NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA
$\mathbf{ord_{\geq i}}$ $\mathbf{lin_i}$ $\mathbf{nodup}$ $\neg ord_{\leq i-1}$ $nsib_{\leq i-1}$ $[\neg no2d]$	$(i > 1)$ NOT-NO2D-NSIB	$[lin_{i-1}]$ $[ord_{\geq i-1}]$ $[\neg ord_{\leq i-2}]$ $[\neg nodup]$	P-PRN ORD-PRN NOT-ORD-UP NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA	$[lin_i]$ $[ord_{\geq i}]$ $[\neg ord_{\leq i-1}]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NSIB-SIB NOT-ORD-FPSIB NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC NOT-NODUP-NO2D-FPA
$\mathbf{ord_{\geq 0}}$ $\mathbf{lin}$ $\neg nodup$ $[lin_1]$	LIN-UP	$[lin]$ $[ord_{\geq 0}]$ $[\neg nodup]$	P-PRN ORD-PRN NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP	$[lin_1]$ $[ord_{\geq 1}]$ $[\neg ord]$ $[\neg nodup]$	P-SIB ORD-SIB NOT-ORD-NODUP-CHLSIB NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ $[\neg nodup]$	NOT-ORD-NODUP-REC NOT-NODUP-STEP

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{ord}_{\geq 1}$ $\mathbf{lin_1}$ $\neg ord$ $\neg nodup$	$[lin]$ P-PRN $[ord_{\geq 0}]$ ORD-PRN $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[lin_1]$ P-SIB $[ord_{\geq 1}]$ ORD-SIB $[\neg ord]$ NOT-ORD-NODUP-CHLSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
$\mathbf{ord}_{\geq \mathbf{i}}$ $(i > 1)$ $\mathbf{lin_i}$ $\neg ord_{\leq i-1}$ $\neg nodup$	$[lin_{i-1}]$ P-PRN $[ord_{\geq i-1}]$ ORD-PRN $[\neg ord_{\leq i-2}]$ NOT-ORD-UP $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[lin_i]$ P-SIB $[ord_{\geq i}]$ ORD-SIB $[\neg ord_{\leq i-1}]$ NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
$\mathbf{ord}_{\geq 0}$ $\mathbf{lin_1}$ $\mathbf{unrel}$ $\mathbf{nodup}$ $nsib$ $[\neg no2d]$ NOT-NO2D-NSIB	$[ord_{\geq 0}]$ ORD-PRN $[lin]$ P-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[ord_{\geq 1}]$ ORD-SIB $[lin_1]$ P-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{ord}_{\geq 0}$ $\mathbf{lin_2}$ $\mathbf{unrel_1}$ $\mathbf{nodup}$ $nsib$ $[unrel]$ UNREL-DOWN $[\neg no2d]$ NOT-NO2D-NSIB	$[ord_{\geq 0}]$ ORD-PRN $[lin_1]$ P-PRN $[unrel]$ P-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[ord_{\geq 1}]$ ORD-SIB $[lin_2]$ P-SIB $[unrel_1]$ P-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{ord}_{\geq 0}$ $(i > 2)$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\mathbf{nodup}$ $nsib$ $[unrel]$ UNREL-DOWN $[\neg no2d]$ NOT-NO2D-NSIB	$[ord_{\geq 0}]$ ORD-PRN $[lin_{i-1}]$ P-PRN $[unrel_{i-2}]$ P-PRN $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[ord_{\geq 1}]$ ORD-SIB $[lin_i]$ P-SIB $[unrel_{i-1}]$ P-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg ord_{\geq 0}]$ NOT-ORD-NO2D-FPA NOT-ORD-NO2D-REC $[\neg nodup]$ NOT-NODUP-NO2D-FPA
$\mathbf{ord}_{\geq 0}$ $\mathbf{lin_1}$ $\mathbf{unrel}$ $\neg nodup$	$[ord_{\geq 0}]$ ORD-PRN $[lin]$ P-PRN $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[ord_{\geq 1}]$ ORD-SIB $[lin_1]$ P-SIB $[\neg ord]$ NOT-ORD-NODUP-CHLSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\mathbf{ord}_{\geq 0}$ $\mathbf{lin_2}$ $\mathbf{unrel_1}$ $\neg nodup$ $[unrel]$ UNREL-DOWN	$[ord_{\geq 0}]$ ORD-PRN $[lin_1]$ P-PRN $[unrel]$ P-PRN $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[ord_{\geq 1}]$ ORD-SIB $[lin_2]$ P-SIB $[unrel_1]$ P-SIB $[\neg ord]$ NOT-ORD-NODUP-CHLSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
$\mathbf{ord}_{\geq 0}$ $(i > 2)$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\neg nodup$ $[unrel]$ UNREL-DOWN	$[ord_{\geq 0}]$ ORD-PRN $[lin_{i-1}]$ P-PRN $[unrel_{i-2}]$ P-PRN $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[ord_{\geq 1}]$ ORD-SIB $[lin_i]$ P-SIB $[unrel_{i-1}]$ P-SIB $[\neg ord]$ NOT-ORD-NODUP-CHLSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
$\mathbf{ord}_{\geq 1}$ $\mathbf{lin_2}$ $\mathbf{unrel_1}$ $\neg ord$ $\neg nodup$ $[unrel]$ UNREL-DOWN	$[ord_{\geq 0}]$ ORD-PRN $[lin_1]$ P-PRN $[unrel]$ P-PRN $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[ord_{\geq 1}]$ ORD-SIB $[lin_2]$ P-SIB $[unrel_1]$ P-SIB $[\neg ord]$ NOT-ORD-NODUP-CHLSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
$\mathbf{ord}_{\geq 1}$ $(i > 2)$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\neg ord$ $\neg nodup$	$[ord_{\geq 0}]$ ORD-PRN $[lin_{i-1}]$ P-PRN $[unrel_{i-2}]$ P-PRN $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[ord_{\geq 1}]$ ORD-SIB $[lin_i]$ P-SIB $[unrel_{i-1}]$ P-SIB $[\neg ord]$ NOT-ORD-NODUP-CHLSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
$\mathbf{ord}_{\geq \mathbf{j}}$ $(i > 2, j > 1)$ $\mathbf{lin_i}$ $\mathbf{unrel_{i-1}}$ $\neg ord_{\leq j-1}$ $\neg nodup$	$[ord_{\geq j-1}]$ ORD-PRN $[lin_{i-1}]$ P-PRN $[unrel_{i-2}]$ P-PRN $[\neg ord_{\leq j-2}$ NOT-ORD-UP $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[ord_{\geq j}]$ ORD-SIB $[lin_i]$ P-SIB $[unrel_{i-1}]$ P-SIB $[\neg ord_{\leq j-1}]$ NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg ord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
(s) $\neg ord_{\geq 0}$ $\neg nodup$	$[negord_{\geq 0}]$ NOT-ORD-UP $[\neg nodup]$ NOT-NODUP-STEP	$[negord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[negord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP	$[negord_{\geq 0}]$ NOT-ORD-NODUP-CHLSIB NOT-ORD-FPSIB $[\neg nodup]$ NOT-NODUP-STEP	$[negord_{\geq 0}]$ NOT-ORD-NODUP-REC $[\neg nodup]$ NOT-NODUP-STEP
(s') $\neg ord_{\geq 0}$ $\neg ntree$ $[nhat]$ NHAT-NTREE $[nsib]$ NSIB-NHAT	$[negord_{\geq 0}]$ NOT-ORD-UP $[\neg ntree]$ NTREE-STEP	$[negord_{\geq 0}]$ NOT-ORD-NSIB-FPA $[\neg ntree]$ NTREE-STEP	$[negord_{\geq 0}]$ NOT-ORD-NSIB-FPA $[\neg ntree]$ NTREE-STEP	$[negord_{\geq 0}]$ NOT-ORD-NSIB-SIB NOT-ORD-FPSIB $[\neg ntree]$ NTREE-STEP	$[negord_{\geq 0}]$ NOT-ORD-NSIB-FPA $[\neg ntree]$ NTREE-STEP

B.2 Correctness of the A_{nodup}^{sloppy} Automaton

		$\downarrow$		$\downarrow^*$		$\downarrow^+$		$\dot{\twoheadrightarrow}$		$\twoheadrightarrow$	
no2d **nodup** $[gen]$ $[no2d_1]$ $[lin]$ $[lin_1]$	 GEN-NO2D NO2D-UP LIN-NO2D LIN-UP	$[gen]$ $[nodup]$ $[nsib]$	GEN-STEP NODUP-CHL NSIB-CHLSIB	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-NO2D-STEP NTREE-SINK NHAT-NTREE NSIB-NHAT NTREE-SINK NOT-UNREL-NTREE	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-NO2D-STEP NTREE-SINK NHAT-NTREE NSIB-NHAT NTREE-SINK NOT-UNREL-NTREE	$[gen]$ $[nodup]$ $[nsib]$	GEN-STEP NODUP-NO2D-STEP NSIB-CHLSIB	$[nodup]$ $[\neg unrel]$	NODUP-NO2D-STEP NTREE-SINK NOT-UNREL-NTREE
gen **nodup** $nsib$ $[\neg no2d]$	 NOT-NO2D-NSIB	$[gen]$ $[nodup]$ $[nsib]$	GEN-STEP NODUP-CHL NSIB-CHLSIB	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-UNREL-DOWN NTREE-SINK NHAT-NTREE NSIB-NHAT NTREE-SINK NOT-UNREL-NTREE	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-UNREL-DOWN NTREE-SINK NHAT-NTREE NSIB-NHAT NTREE-SINK NOT-UNREL-NTREE	$[\neg nodup]$	NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$	NOT-NODUP-NO2D-FPA
nodup $nsib$ $\neg unrel$ $[\neg no2d]$	 NOT-NO2D-UNREL	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-CHL NSIB-CHLSIB NOT-UNREL-PRNCHL	$[\neg nodup]$	NOT-NODUP-UNREL-DSC	$[\neg nodup]$	NOT-NODUP-UNREL-DSC	$[\neg nodup]$	NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$	NOT-NODUP-NO2D-FPA
lin **nodup** $\neg unrel$ $[lin_1]$ $[\neg no2d]$ $[nolc]$ $[norc]$	 LIN-UP NOT-NO2D-UNREL NOLC-LIN NORC-LIN	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-CHL NSIB-CHLSIB NOT-UNREL-PRNCHL	$[\neg nodup]$	NOT-NODUP-UNREL-DSC	$[\neg nodup]$	NOT-NODUP-UNREL-DSC	$[unrel]$ $[nodup]$ $[nsib]$	UNREL-NOLC-FLS NODUP-LIN-PRNSIB NSIB-CHLSIB	$[\neg nodup]$	NOT-NODUP-NO2D-FPA
unrel **nodup** $nsib$ $[\neg no2d]$	 NOT-NO2D-NSIB	$[unrel]$ $[nodup]$ $[nsib]$	P-CHL UNREL-DOWN NODUP-CHL NSIB-CHLSIB	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-UNREL-DOWN NTREE-SINK NHAT-NTREE NSIB-NHAT NTREE-SINK NOT-UNREL-NTREE	$[nodup]$ $[nsib]$ $[\neg unrel]$	NODUP-UNREL-DOWN NTREE-SINK NHAT-NTREE NSIB-NHAT NTREE-SINK NOT-UNREL-NTREE	$[\neg nodup]$	NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$	NOT-NODUP-NO2D-FPA
$\neg nodup$		$[\neg nodup]$	NOT-NODUP-STEP	$[\neg nodup]$	NOT-NODUP-STEP	$[\neg nodup]$	NOT-NODUP-STEP	$[\neg nodup]$	NOT-NODUP-STEP	$[\neg nodup]$	NOT-NODUP-STEP

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
no2d **nodup** $[gen]$ GEN-NO2D $[no2d_1]$ NO2D-UP $[lin]$ LIN-NO2D $[lin_1]$ LIN-UP	$[no2d]$ P-PRN $[nodup]$ NODUP-NO2D-STEP	$[lin]$ LIN-AOS $[nodup]$ NODUP-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC	$[lin]$ LIN-ANC $[nodup]$ NODUP-NO2D-STEP $[\neg unrel]$ NOT-UNREL-ANC	$[gen]$ GEN-STEP $[nodup]$ NODUP-NO2D-STEP $[nsib]$ NSIB-CHLSIB	$[nodup]$ NODUP-NO2D-STEP $[\neg unrel]$ NTREE-SINK NOT-UNREL-NTREE
gen **nodup** *nsib* $[\neg no2d]$ NOT-NO2D-NSIB	$[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA
nodup *nsib* $\neg unrel$ $[\neg no2d]$ NOT-NO2D-UNREL	$[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA
lin **nodup** $\neg unrel$ $[lin_1]$ LIN-UP $[\neg no2d]$ NOT-NO2D-UNREL $[nolc]$ NOLC-LIN $[norc]$ NORC-LIN	$[lin]$ P-PRN $[nodup]$ NODUP-LIN-PRNSIB $[\neg unrel]$ NOT-UNREL-PRNCHL	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[unrel]$ UNREL-NORC-PRS $[nodup]$ NODUP-LIN-PRNSIB $[nsib]$ NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA
unrel **nodup** *nsib* $[\neg no2d]$ NOT-NO2D-NSIB	$[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[\neg nodup]$ NOT-NODUP-NO2D-FPA	$[lin_2]$ P-SIB $[nolc_1]$ P-SIB $[norc_1]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nsib_1]$ NP-SIB $[nhat]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB	$[\neg nodup]$ NOT-NODUP-NO2D-FPA

Continued on next page

Continued from previous page

	$\downarrow$	$\downarrow^*$	$\downarrow^+$	$\dot{\twoheadrightarrow}$	$\twoheadrightarrow$
$\neg nodup$	$[\neg nodup]$ NOT-NODUP-STEP	$[\neg nodup]$ NOT-NODUP-STEP	$[\neg nodup]$ NOT-NODUP-STEP	$[lin_i]$ P-SIB $[nolc_{i-1}]$ P-SIB $[norc_{i-1}]$ P-SIB $[\neg no2d_{\geq 0}]$ NOT-NO2D-STEP $[\neg unrel]$ NOT-UNREL-NORC-PRS $[nsib_{i-1}]$ NP-SIB $[nhat_{\leq i-2}]$ NHAT-SIB $[\neg ord]$ NOT-ORD-NSIB-SIB $[\neg nodup]$ NOT-NODUP-NSIB-PRNSIB $[\neg nodup]$ NOT-NODUP-STEP	$[\neg nodup]$ NOT-NODUP-STEP

Bibliography

[1] M. Benedikt, W. Fan, and G. Kuper. Structural properties of XPath fragments. In *Proc. of ICDT'03, 2003.*, 2003.

[2] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XML Path Language (XPath) 2.0, W3C Working Draft, Nov 2003. `http://www.w3.org/TR/xpath20`.

[3] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0 An XML Query Language, W3C Working Draft, Nov 2003. `http://www.w3.org/TR/xquery`.

[4] B. Choi, M. Fernández, and J. Siméon. The XQuery Formal Semantics: A Foundation for Implementation and Optimization. Internal Working Document at AT&T and Lucent. `http://www.cis.upenn.edu/~kkchoi/galax.pdf`, 2002.

[5] D. Draper, P. Fankhauser, M. Fernández, A. Malhotra, K. Rose, M. Rys, J. Siméon, and P. Wadler. XQuery 1.0 and XPath 2.0 formal semantics, W3C Working Draft, Feb 2004. `http://www.w3.org/TR/query-semantics`.

[6] M. Fernández and J. Siméon. *Galax, the XQuery implementation for discriminating hackers.* Lucent Technologies – Bell Labs, v0.3 edition, 2003. `http://www-db-out.bell-labs.com/galax`.

[7] M. Fernandez, J. Siméon, B. Choi, A. Marian, and G. Sur. Implementing XQuery 1.0: The Galax Experience. In *Proceedings of International Conference on Very Large Databases (VLDB)*, pages 1077–1080, Berlin, Germany, Sept. 2003.

[8] A. GoldBerg and R. Paige. Stream Processing. In *Proceedings of the ACM Symposium on LISP and Functional Programming Languages*, pages 53–62, 1984.

[9] G. Gottlob, C. Koch, and R. Pichler. Efficient Algorithms for Processing XPath Queries. In *Proc. of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, Hong Kong, 2002.

[10] G. Gottlob, C. Koch, and R. Pichler. The Complexity of XPath Query Evaluation. In *Proc. of the 22nd ACM SIGACT-SIGMOD-GIGART Symposium on Priciples of Database Systems (PODS)*, San Diego (CA), 2003.

[11] T. Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120, Madison, 2002.

[12] T. Grust, M. van Keulen, and J. Teubner. Staircase Join: Teach a Relational DBMS to Watch its (Axis) Steps. In *VLDB 2003*, 2003.

[13] S. Helmer, C.-C. Kanne, and G. Moerkotte. Optimized Translation of XPath into Algebraic Expressions Parameterized by Programs Containing Navigational Primitives. In *Proc. of the 3rd International Conference on Web Information Systems Engineering (WISE 2002)*, pages 215–224, Singapore, 2002.

[14] J. Hidders and P. Michiels. Efficient xpath axis evaluation for dom data structures. In *PLAN-X*, Venice, Italia, 2004.

[15] M. Kay. XSL transformations (XSLT) version 2.0, 2003. `http://www.w3.org/TR/xslt20`.

[16] A. Kwong and M. Gertz. Schema-based optimization of XPath expressions. Technical report, Univ. of California, dept. of Computer Science, 2001.

[17] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon). *The Objective Caml system, Documentation and user's manual*, release 3.07 edition, 2003. `http://caml.inria.fr/ocaml/htmlman`.

[18] F. Peng and S. S. Chawathe. Xpath queries on streaming data. In *Proc. of the 22nd ACM SIGACT-SIGMOD-GIGART Symposium on Priciples of Database Systems (PODS)*, San Diego (CA), 2003.

[19] A. Schmidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, and R. Busse. XMark: A benchmark for XML data management. In *Proceedings of International Conference on Very Large Databases (VLDB)*, pages 974–985, Hong Kong, China, Aug. 2002. `http://monetdb.cwi.nl/xml/`.