Chapter 6

Expressive power of CoreXPath restricted to the descendant relation

XML is a standard for storage and exchange of data on the internet. Its basic data structure is that of finite sibling-ordered tree: a finite tree in which the children of each node are linearly ordered. Several languages were introduced to describe XML documents and among them, the language XPath, which is particularly convenient for selecting nodes and describing paths. In order to be able to study XPath from a logical point of view, Georg Gottlob, Christoph Koch and Reinhard Pichler isolated an essential navigational core of XPath [GKP05], called CoreXPath.

The logic CoreXPath is essentially a modal logic and the XML documents are nothing but Kripke models with two basic modalities (one for the child relation and the other one for the relation between the siblings of a node). The main difference between CoreXPath and modal logic is that CoreXPath is a two-sorted language: it contains both nodes expressions (which would be similar to formulas, in the sense that they select points in a tree) and path expressions (which are like PDL programs, as they select paths in a tree).

In this chapter, we exploit the connection between CoreXPath and modal logic. The goal is not so much to prove very technical theorems, but to illustrate how, by combining well-chosen results of modal logic, we can easily obtain results for CoreXPath. Moreover, one of the results of the modal logic area that we use, is an easy consequence of Theorem 5.5.3, established in the last chapter.

The results that we present, concerns the expressive power of CoreXPath. It is easy to prove that CoreXPath is a fragment of first-order logic (using a variant of the standard translation, presented in Chapter 2). However, not all first order formulas (over the appropriate signature) are expressible in CoreXPath. In fact, it was shown by Maarten Marx and Maarten de Rijke [MdR05] that the CoreXPath node expressions capture exactly the two-variable first order formulas with one and two free variables. A characterization in the same fashion for CoreXPath path expression was also obtained.
In this chapter, we focus on CoreXPath(↓+); that is the fragment of CoreXPath for which the only axis allowed corresponds to the descendant relation (or to put it in terms of modal logic, the only modality considered is associated with the descendant relation). In determining the expressive power of this language, there are at least two natural yardsticks. One is first-order logic, which is probably the most well known logical language. The second, even more attractive one, is monadic second order logic, which is a very well-behaved language on trees (see Chapter 2). As mentioned earlier, CoreXPath is a fragment of first-order logic. However, if we can characterize a fragments of CoreXPath in terms of $\text{MSO}$, this means that we have a stronger result (in the sense that we can immediately derive a characterization of this fragment in terms of $\text{FO}$).

Our two main results are a characterization of CoreXPath(↓+) node expressions and a characterization of CoreXPath(↓+) path expressions, both in terms of $\text{MSO}$. Each characterization is expressed in two different ways: using bisimulations and in terms of simple operations on trees. Moreover, we can derive from these results a decision procedure for establishing whether a given $\text{MSO}$ formula is equivalent to a CoreXPath(↓+) node expression or path expression.

The proofs of both characterizations follow the same scheme. Each of them consists in combining two results concerning the $\mu$-calculus. In the case of the characterization for node expressions, the first result is the Janin-Walukiewicz theorem (which characterizes the $\mu$-calculus as a fragment of $\text{MSO}$) and the second result is a consequence of the de Jongh-fixpoint theorem, mentioned in [Ben06] (which says that on conversely well-founded transitive models, the $\mu$-calculus and modal logic have the same expressive power). In the case of the characterization for path expressions, we use adaptations of these two results for $\mu$-programs. The adaptation of the Janin-Walukiewicz theorem for $\mu$-programs is a direct consequence of Theorem 5.5.3.

For CoreXPath(↓+) node expressions, similar characterizations have already been proved by Mikołaj Bojańczyk and Igor Walukiewicz [BW06, BW07], using the framework of forest algebras (and without reference to modal logic). The logic EF in their work corresponds to the node expressions of CoreXPath(↓+). Mikołaj Bojańczyk and Igor Walukiewicz [BW06] also established a similar characterization for the fragment of CoreXPath where the only axis, or modalities, allowed corresponds to the child relation and the descendant relation. Let us finally mention that Mikołaj Bojańczyk [Boj07] found a characterization of the fragment of CoreXPath using the axis (modalities) associated to the ancestor and the descendant relations.

Other alternative proofs for the characterization of CoreXPath(↓+) node expressions have also appeared, see e.g. [Él08], [Wu07] and [DO09]. The proof presented here has been found independently and uses different methods (combining well-known results from modal logic and $\mu$-calculus). The advantage of this new proof is that it can easily be extended to a similar characterization for path expressions.
6.1 Preliminaries

6.1.1 XML trees

The language XPath is based on a tree representation of the XML documents. Formally, given an infinite set $\text{Prop}$ of proposition letters, we define an XML tree as a structure $T = (W, R, R_{\rightarrow}, V)$, where

- $(W, R)$ is a finite tree (with $W$ the set of nodes and $R$ the child relation),
- $R_{\rightarrow}$ is the successor relation of some linear ordering between siblings in the tree,
- $V : \text{Prop} \rightarrow \mathcal{P}(W)$ labels the nodes with elements of $\text{Prop}$.

So an XML tree is nothing but a Kripke model for a modal language with two modalities: one corresponding to the child relation and the other one to the next sibling relation. In this particular setting, the elements of $\text{Prop}$ correspond to XML tags, such as, in the case of HTML, $\text{body}$, $\text{p}$, $\text{it}$, .... It is customary to require that each node satisfies precisely one tag. In order to simplify the presentation, it will be convenient for us not to make this requirement from the start. However, all results can be adapted to the setting with unique node labels.

6.1.2 CoreXPath, the navigational core of XPath 1.0

There are two main types of expressions in CoreXPath: path expressions and node expressions. Path expressions describe ways of traveling through the tree and they are interpreted as binary relations, while node expressions are used to describe properties of nodes and are interpreted as subsets. More precisely, the syntax of CoreXPath is defined as follows:

$$\begin{align*}
\text{Step} & := \downarrow | \leftarrow | \uparrow | \rightarrow, \\
\text{Axis} & := \text{Step} | \text{Step}^+, \\
\text{PathEx} & := . | \text{Axis} | \text{PathEx} [\text{NodeEx}] | \text{PathEx}/\text{PathEx} | \text{PathEx} \cup \text{PathEx}, \\
\text{NodeEx} & := p | \langle \text{PathEx} \rangle | \neg \text{NodeEx} | \text{NodeEx} \lor \text{NodeEx},
\end{align*}$$

where $p$ belongs to $\text{Prop}$.

The axes correspond to basic moves one can make in the tree. The axe “.” corresponds to staying at the current node. The axes $\downarrow$, $\leftarrow$, $\uparrow$ and $\rightarrow$ correspond
respectively to the child relation, the next sibling relation, the parent relation and the previous sibling relation. Moreover, given one of these four axes $A$, the axe $A^+$ corresponds to the transitive closure of the relation associated to $A$. The axes can be composed into path expressions by using composition $(;)$, union $(\cup)$, and node tests $(\cdot\cdot\cdot)$. The node expression $(\text{PathEx})$ expresses that the current node belongs to the domain of the binary relation defined by PathEx.

The reader familiar with original XPath notation will notice that we included a number of abbreviations and alterations. Table 6.1 provides a comparison of our notation with that of [W3C].

The semantics of CoreXPath expressions is given by two functions, $\llbracket \cdot \rrbracket_{\text{PExpr}}$ and $\llbracket \cdot \rrbracket_{\text{NExpr}}$. For any path expression $A$ and XML tree $T$, $\llbracket A \rrbracket_{\text{PExpr}}$ denotes a binary relation over the domain of $T$, and for any node expression $\varphi$ and XML tree $T$, $\llbracket \varphi \rrbracket_{\text{NExpr}}$ denotes a subset of the domain of $T$. Given an XML tree $T = (T, R, R_{\rightarrow}, V)$, the binary relation $\llbracket \cdot \rrbracket_{\text{PExpr}}$ and the subset $\llbracket \cdot \rrbracket_{\text{NExpr}}$ are defined by induction in the following way:

\[
\begin{align*}
\llbracket \cdot \rrbracket_{\text{PExpr}} &= \{(u, u) \mid u \in T\}, \\
\llbracket a \rrbracket_{\text{PExpr}} &= R_a \text{ for all } a \in \text{Step}, \\
\llbracket a^+ \rrbracket_{\text{PExpr}} &= (R_a)^+ \text{ for all } a \in \text{Step}, \\
\llbracket A/B \rrbracket_{\text{PExpr}} &= \{(u, v) \mid \exists w \text{ such that } (u, w) \in \llbracket A \rrbracket_{\text{PExpr}} \text{ and } (w, u) \in \llbracket B \rrbracket_{\text{PExpr}}\}, \\
\llbracket A \cup B \rrbracket_{\text{PExpr}} &= \llbracket A \rrbracket_{\text{PExpr}} \cup \llbracket B \rrbracket_{\text{PExpr}}, \\
\llbracket A \mid \varphi \rrbracket_{\text{PExpr}} &= \{(u, v) \mid (u, v) \in \llbracket A \rrbracket_{\text{PExpr}} \text{ and } v \in \llbracket \varphi \rrbracket_{\text{NExpr}}\}, \\
\llbracket p \rrbracket_{\text{NExpr}} &= \{u \mid u \in V(p)\}, \\
\llbracket (\text{PathEx}) \rrbracket_{\text{NExpr}} &= \{u \mid \exists v \text{ such that } (u, v) \in \llbracket \text{PathEx} \rrbracket_{\text{PExpr}}\}, \\
\llbracket \neg \varphi \rrbracket_{\text{NExpr}} &= \{u \mid u \notin \llbracket \varphi \rrbracket_{\text{NExpr}}\}, \\
\llbracket \varphi \vee \psi \rrbracket_{\text{NExpr}} &= \llbracket \varphi \rrbracket_{\text{NExpr}} \cup \llbracket \psi \rrbracket_{\text{NExpr}},
\end{align*}
\]

where $R_\downarrow$ is the relation $R$, $R_{\leftarrow}$ is the converse of the relation $R$, $R_\uparrow$ is the converse of the relation $R$ and given a binary relation $R_a$, $(R_a)^+$ is the transitive closure of $R_a$. For readability, the superscript $T$ is left out.
For $A \subseteq \text{Axis}$, we will denote by CoreXPath($A$) the fragment of CoreXPath in which the only allowed axes are those listed in $A$.

### 6.1.3 Connections with modal logic

There are two main differences between modal logic and CoreXPath. First, the semantics for CoreXPath is more restrictive (finite sibling-ordered trees as opposed to arbitrary Kripke structure). Next, the syntax for CoreXPath is two-sorted (node and path expressions, the interpretations of which are respectively subsets and binary relations), whereas in modal logic, only formulas (which are interpreted as subsets) are considered. However, we can easily obtain a two-sorted syntax for modal logic, by introducing modal programs. The definition of modal program is a simplified version of the notion of program for PDL.

**Modal programs** Given a set $A$ of actions, we define the set of modal programs over the set $A$ of actions by induction in the following way:

$$\theta ::= R_a \mid \varphi? \mid \theta; \theta \mid \theta \cup \theta,$$

where $a \in A$ and $\varphi$ is a modal formula over the set $A$ of actions.

Given a Kripke model $\mathcal{M}$, the interpretation of a modal program $\theta$ is a binary relation $[\theta]_{\mathcal{M}}$ over the domain of the model. This interpretation is defined by induction on the complexity of the program. We do not give more details, as this definition is a particular case of the semantics for PDL (see Chapter 2) and the semantics for the $\mu$-programs (see Chapter 2). We only recall that the interpretation of $\varphi?$ is the relation $\{(u, u) \mid \varphi \text{ is true at } u\}$.

We would like to observe that the syntax for modal programs is the same as the one for PDL, except that we do not use the Kleene star and that we can only test with modal formulas (instead of PDL formulas). Moreover, in PDL, we also allow formulas of the form $(\theta)\varphi$ (where $\theta$ is a program and $\varphi$ a formula) and it is not the case for modal formulas. However, it is possible to show that for all modal programs $\varphi$ and for all modal programs, $(\theta)\varphi$ is equivalent to a modal formula (in fact, this can be proved easily using the translation $\tau_2$ from the proof of the next proposition).

**Equivalence between CoreXPath and modal logic** Let $A$ be a subset of $\text{Axis}$. Given a CoreXPath($A$) node expression $\varphi$ and a modal formula over $A$, we say that $\varphi$ and $\psi$ are equivalent (on finite trees) if for all XML trees $T = (T, R, R_\rightarrow)$, we have $[\varphi]_{T}^\text{NEexpr} = [\psi]_{T}$.

Similarly, given a CoreXPath($A$) path expression $A$ and a modal program $\theta$ over $A$, $A$ and $\theta$ are equivalent (on finite trees) if for all XML trees $T = (T, R, R_\rightarrow)$, we have $[A]_{T}^\text{PEexpr} = [\theta]_{T}$. 
6.1.1. Proposition. Let $A$ be a subset of Axis. There is an effective truth-preserving translation from the set of CoreXPath($A$) node and path expressions to the set of modal formulas and programs over $A$, and vice-versa.

Proof Let $A$ be a subset of Axis. Both translations are very similar and not difficult to define; so we only give details for the translation which maps a CoreXPath($A$) node expression to an equivalent modal formula (over $Prop$ and $A$) and maps a CoreXPath($A$) path expression to an equivalent modal program (over $Prop$ and $A$). This translation $\tau$ is defined as the composition of two translations $\tau_1$ and $\tau_2$, which we define below.

The translation $\tau_1$ is defined by induction on the complexity of the path and node expressions as follows:

$$
\begin{align*}
\tau_1(a) &= R_a, \\
\tau_1(A[\varphi]) &= \pi_1(A); \tau_1(\varphi), \\
\tau_1(A \cup B) &= \tau_1(A) \cup \tau_1(B), \\
\tau_1((A)) &= \langle \tau_1(A) \rangle^?, \\
\tau_1(\varphi \lor \psi) &= \tau_1(\varphi) \lor \tau_1(\psi),
\end{align*}
$$

where $a$ belongs to $A$, $A$ and $B$ are CoreXPath($A$) path expressions, $\varphi$ and $\psi$ are CoreXPath($A$) node expressions and $p$ is a proposition letter. Note that $\tau_1$ does not necessarily map a node expression to a modal formula: formulas of the form $\langle \theta \rangle \varphi$ might occur in the range of $\tau_1$. Such formulas are not modal formulas, as defined in Chapter 2.

To fix this problem, we introduce a translation $\tau_2$ which is defined by induction on the complexity of the formulas and programs in the range of $\tau_1$:

$$
\begin{align*}
\tau_2(R_a) &= R_a, \\
\tau_2(\varphi^?) &= \tau_2(\varphi)^?, \\
\tau_2(\theta; \lambda) &= \tau_2(\theta); \tau_2(\lambda), \\
\tau_2(\theta \cup \lambda) &= \tau_2(\theta) \cup \tau_2(\lambda), \\
\tau_2(p) &= p, \\
\tau_2(\neg p) &= \neg p, \\
\tau_2(\varphi \lor \psi) &= \tau_2(\varphi) \lor \tau_2(\psi), \\
\tau_2(\varphi \land \psi) &= \tau_2(\varphi) \land \tau_2(\psi), \\
\tau_2(\langle \theta \rangle \varphi) &= \Diamond_a \tau_2(\varphi), \\
\tau_2(\langle \theta \rangle^? \psi) &= \tau_2(\varphi) \land \tau_2(\psi), \\
\tau_2(\langle \theta; \lambda \rangle \varphi) &= \tau_2(\langle \theta \rangle \tau_2((\lambda) \varphi)), \\
\tau_2(\langle \theta \cup \lambda \rangle \varphi) &= \tau_2(\langle \theta \rangle \varphi) \lor \tau_2((\lambda) \varphi),
\end{align*}
$$

where $a$ belongs to $A$, $p$ is a proposition letter, $\varphi$ and $\psi$ are PDL formulas in the range of $\tau_1$, $\theta$ and $\lambda$ are PDL programs in the range of $\tau_1$.

Finally we define $\tau$ as $\tau_2 \circ \tau_1$. It is easy to check that $\tau$ has the required properties.

As mentioned in the introduction, throughout the chapter, this connection will help us to transfer well-known results of the modal logic area into the framework of CoreXPath.
6.2 CoreXPath(↓⁺) node expressions

We start by characterizing the CoreXPath(↓⁺) node expressions as a fragment of monadic second order logic. Two characteristic features of CoreXPath(↓⁺) are that (i) whether a node expression holds at a node depends only on the subtree below it, and (ii) CoreXPath(↓⁺) expressions cannot see the difference between children and descendants. It turns out that, in some sense, these two properties characterize CoreXPath(↓⁺) as a fragment of monadic second-order logic. We formalize these two features in two ways: using transitive bisimulations and in terms of simple operations on trees.

Before we state the characterization, we fix some notation and introduce some terminology.

Convention As the only axe that we consider is ↓⁺, we can forget about the sibling order. More precisely, instead of evaluating node and path expressions on XML trees, we interpret these expressions on finite tree models (as defined in Section 2.6). Moreover, in this chapter, we never consider frames. So there is no confusion if we use the word “tree” instead of “tree model” and we will do so throughout this chapter.

Finally, recall that when we talk about MSO formulas on models, we always have the same signature in mind, which consists of a binary relation and a unary predicate for each proposition letter (for more details, see Section 2.6). In particular, the binary relation corresponds to the child relation, when we interpret an MSO formula on a tree.

Equivalence An MSO formula φ(x) is equivalent to a CoreXPath(↓⁺) node expression ψ if for all finite trees T and all nodes u ∈ T, we have T, u ⊨ φ(x) iff u belongs to [ψ]T. When this happens, we write φ(x) ≡ ψ.

Transitive bisimulation Let M = (W, R, V) and M′ = (W′, R′, V′) be two Kripke models. A relation B ⊆ W × W′ is a transitive bisimulation if for all (w, w′) in B, we have

- the same proposition letters hold at w and w′,
- if wR⁺v, there exists v′ ∈ W′ such that w′(R′)⁺v′ and (v, v′) ∈ B,
- if w′(R′)⁺v′, there exists v ∈ W such that wR⁺v and (v, v′) ∈ B.

A transitive bisimulation B between two models M = (W, R, V) and M′ = (W′, R′, V′) is total if the domain of B is W and the range of B is W′.

A MSO formula φ(x) with one free first order variable is said to be invariant under (transitive) bisimulation on a class C of models if for all models M, M′ in
Chapter 6. CoreXPath restricted to the descendant relation

Figure 6.1: The trees $\mathcal{T}$ and $\text{COPY}_{u \rightarrow v}(\mathcal{T})$.

$\mathcal{C}$, all (transitive) bisimulations $B \subseteq \mathcal{M} \times \mathcal{M}'$, and pairs $(w, w') \in B$, we have $\mathcal{M}, w \models \varphi(x)$ iff $\mathcal{M}', w' \models \varphi(x)$.

Intuitively, a transitive bisimulation is nothing but a regular bisimulation, except that instead of considering the successor relation $R$, we focus on the transitive closure of $R$.

The operation copy Let $\mathcal{T} = (T, R, V)$ be a finite tree and let $u, v$ be nodes such that $v$ is a descendant of $u$. Recall that $\mathcal{T}_v$ is the submodel of $\mathcal{T}$ generated by $v$. We write $\text{COPY}(\mathcal{T}_v)$ for a tree that is an isomorphic copy of $\mathcal{T}_v$. In order to make a distinction between a point $w$ in $\mathcal{T}_v$ and the copy of $w$ in $\text{COPY}(\mathcal{T}_v)$, we denote by $c(w)$ the copy of $w$. We define $\text{COPY}_{u \rightarrow v}(\mathcal{T})$ as the tree that is obtained by adding the isomorphic copy $\text{COPY}(\mathcal{T}_v)$ to the tree $\mathcal{T}$, plus an edge from $u$ to the copy $c(v)$ of $v$, see Figure 6.1.

This definition allows us to make precise what it means not to distinguish children from descendants: it means that $\mathcal{T}$ and $\text{COPY}_{u \rightarrow v}(\mathcal{T})$ are indistinguishable.

Invariance under the subtree and the copy operations Let $\varphi(x)$ be an MSO formula. We say that $\varphi(x)$ is invariant under the subtree operation (on finite trees) if for all finite trees $\mathcal{T}$ and nodes $u$,

$$\mathcal{T}, u \models \varphi(x) \iff \mathcal{T}_u, u \models \varphi(x).$$

The formula $\varphi(x)$ is invariant under the copy operation (on finite trees) if for all finite trees $\mathcal{T}$ with root $r$, and with nodes $u, v$ such that $v$ is a descendant of $u$,

$$\mathcal{T}, r \models \varphi(x) \iff \text{COPY}_{u \rightarrow v}(\mathcal{T}), r \models \varphi(x).$$

We can now state the characterization of CoreXPath$(\downarrow^+)$ precisely.

6.2.1. Theorem. Let $\varphi(x)$ be an MSO formula. The following are equivalent:

(i) $\varphi(x)$ is equivalent to a CoreXPath$(\downarrow^+)$ node expression,
6.2. CoreXPath(\downarrow^+) node expressions

(ii) \( \varphi(x) \) is invariant under transitive bisimulation on finite trees,

(iii) \( \varphi(x) \) is invariant under the subtree and the copy operations.

Moreover, for all MSO formulas \( \varphi(x) \), we can compute a CoreXPath(\downarrow^+) node expression \( \psi \) such that \( \varphi(x) \equiv \psi \) iff \( \varphi(x) \) is equivalent to a CoreXPath(\downarrow^+) node expression.

The proof will be based on two known expressivity results. The first theorem we use is the bisimulation characterization of the modal \( \mu \)-calculus, due to David Janin and Igor Walukiewicz (see Section 2.6 and [JW96]). This characterization works on arbitrary Kripke models, but also on the restricted class of finite trees, which is important for us.

Moreover, in the case of finite trees, the characterization is effective: it is decidable whether an MSO formula is invariant under bisimulation, and for bisimulation invariant MSO formulas an equivalent formula of the modal \( \mu \)-calculus can be effectively computed. Recall that an MSO formula \( \varphi(x) \) is equivalent on finite trees, to a \( \mu \)-sentence \( \psi \) if for all finite trees \( T \) and nodes \( u \in T \), \( T, u \models \varphi(x) \) iff \( T, u \models \psi \).

6.2.2. Theorem (from [JW96]). An MSO formula \( \varphi(x) \) is equivalent on finite trees to a \( \mu \)-sentence iff \( \varphi(x) \) is invariant under bisimulation on finite trees.

Moreover, for all MSO formulas \( \varphi(x) \), we can compute a \( \mu \)-sentence \( \psi \) such that \( \varphi(x) \) and \( \psi \) are equivalent on finite trees iff \( \varphi(x) \) is equivalent on finite trees to a \( \mu \)-sentence.

Proof Let \( \varphi(x) \) be an MSO formula. It is immediate that if \( \varphi(x) \) is equivalent on finite trees to a \( \mu \)-sentence, then \( \varphi(x) \) is invariant under bisimulation on finite trees. For the other direction of the implication, suppose that \( \varphi(x) \) is invariant under bisimulation on finite trees.

It follows from the proof of the main result of [JW96] (Theorem 11) that we can compute a \( \mu \)-sentence \( \psi \) such that for all pointed models \( (M, w) \), we have

\[ M, w \models \psi \iff M_\omega^w, w \models \varphi(x). \]

Recall that \( M_\omega^w \) is the \( \omega \)-expansion of the pointed model \( (M, w) \) (see Section 2.6).

A careful inspection of the proof shows that we can even get a stronger result: we can compute \( n \in \mathbb{N} \) and \( \mu \)-sentence \( \psi \) such that for all pointed models \( (M, w) \), we have

\[ M, w \models \psi \iff (M, w)^n, w \models \varphi(x). \]

Recall that \( (M, w)^n \) is the \( n \)-expansion of the pointed model \( (M, w) \) (see Section 2.6). In particular, for all finite trees \( T \) and for all \( u \in T \), we have

\[ T, u \models \psi \iff (T, u)^n, u \models \varphi(x). \] (6.1)
Now recall that there is a bisimulation \( B \) between \( (\mathcal{T}, u)^n \) and \( \mathcal{T} \) such that \( (u, u) \in B \) (see Section 2.6). Since \( \varphi(x) \) is invariant under bisimulation on finite trees, this implies that
\[
(\mathcal{T}, u)^n, u \vDash \varphi(x) \iff \mathcal{T}, u \vDash \varphi(x).
\]
Putting this together with (6.1), we obtain that
\[
\mathcal{T}, u \vDash \psi \iff \mathcal{T}, u \vDash \varphi(x).
\]
Therefore, \( \varphi(x) \) is equivalent on finite trees to a \( \mu \)-sentence.

Moreover, it also easily follows from our proof that given an MSO formula \( \varphi(x) \), we can compute a \( \mu \)-sentence \( \psi \) such that \( \varphi(x) \) and \( \psi \) are equivalent on finite trees iff \( \varphi(x) \) is equivalent on finite trees to a \( \mu \)-sentence. \( \square \)

The second result we use is a consequence of the de Jongh-fixpoint theorem which was proved independently by Dick de Jongh and Giovanni Sambin (see [Smo85]). More specifically, we use the fact that the \( \mu \)-calculus over models for Gödel-Löb logic (or equivalently, evaluated over transitive Kripke models, that do not contain any infinite path) collapses to its modal fragment, as was first observed by Johan van Benthem in [Ben06].

6.2.3. Theorem ([Ben06]). For all \( \mu \)-sentences \( \varphi \), we can compute a modal formula \( \psi \) satisfying the following: For all Kripke models \( \mathcal{M} = (W, R, V) \) such that \( R \) is transitive and \( \mathcal{M} \) does not contain any infinite path, for all \( w \in W \), we have \( \mathcal{M}, w \models \varphi \iff \mathcal{M}, w \models \psi \).

In particular, for all finite transitive trees \( \mathcal{T}^+ \) and all nodes \( u \in \mathcal{T}^+ \), we have \( \mathcal{T}^+, u \models \varphi \iff \mathcal{T}^+, u \models \psi \).

We are now ready to prove that CoreXPath(\( \downarrow^+ \)) is the transitive bisimulation invariant fragment of MSO, by putting together Theorem 6.2.2 and Theorem 6.2.3.

6.2.4. Proposition. An MSO formula \( \varphi(x) \) is equivalent to a CoreXPath(\( \downarrow^+ \)) node expression iff \( \varphi(x) \) is invariant under transitive bisimulation on finite trees.

Moreover, for all MSO formulas \( \varphi(x) \), we can compute a CoreXPath(\( \downarrow^+ \)) node expression \( \psi \) such that \( \varphi(x) \equiv \psi \iff \varphi(x) \) is equivalent to a CoreXPath(\( \downarrow^+ \)) node expression.

Proof First we show that an MSO formula \( \varphi(x) \) is equivalent to a CoreXPath(\( \downarrow^+ \)) node expression iff \( \varphi(x) \) is invariant under transitive bisimulation on finite trees. We restrict ourselves to prove the difficult direction (the other one is a standard induction on the complexity of CoreXPath(\( \downarrow^+ \)) node expressions). Let \( \varphi(x) \) be a MSO formula that is invariant under transitive bisimulation on finite trees. We need to find a node expression \( \chi \) of CoreXPath(\( \downarrow^+ \)) such that for all finite trees \( \mathcal{T} \) and all nodes \( u \), \( \chi \) holds at \( u \) iff \( \varphi(u) \) is true.
By Theorem 6.2.2, we can compute a \( \mu \)-sentence \( \psi \) such that \( \varphi(x) \) and \( \psi \) are equivalent on finite trees iff \( \varphi(x) \) is invariant under bisimulation. Since \( \varphi(x) \) is invariant under transitive bisimulation on finite trees, in particular \( \varphi(x) \) is invariant under ordinary bisimulation on finite trees. Hence, \( \varphi(x) \) and \( \psi \) are equivalent on finite trees.

Now we show that for all finite trees \( \mathcal{T} = (T, R, V) \) and all nodes \( u \) in \( \mathcal{T} \), we have

\[
\mathcal{T}, u \vDash \psi \iff \mathcal{T}^+, u \vDash \psi, \quad (6.2)
\]

where \( \mathcal{T}^+ = (T, R^+, V) \). Take a finite tree \( \mathcal{T} = (T, R, V) \) and a node \( u \) in \( \mathcal{T} \). Let \( \mathcal{T}^+ \) be the model \( (T, R^+, V) \) and let \( \mathcal{S} \) be the unraveling of the pointed model \( (\mathcal{T}^+, u) \) (see Section 2.6).

The canonical bisimulation between \( \mathcal{S} \) and \( \mathcal{T}^+ \) links \( u \) in \( \mathcal{S} \) with \( u \) in \( \mathcal{T}^+ \). It follows that \( \mathcal{T}^+, u \vDash \psi \iff \mathcal{S}, u \vDash \psi \). Moreover, the canonical bisimulation between \( \mathcal{S} \) and \( \mathcal{T}^+ \) constitutes a transitive bisimulation between \( \mathcal{S} \) and \( \mathcal{T} \), which links the node \( u \) in \( \mathcal{S} \) to the node \( u \) in \( \mathcal{T} \). Since \( \psi \) is equivalent to \( \varphi(x) \) on infinite trees and \( \varphi(x) \) is invariant under transitive bisimulation on finite trees, we have that

\[
\mathcal{S}, u \vDash \psi \iff \mathcal{T}, u \vDash \psi.
\]

This finishes the proof of (6.2).

Next it follows from Theorem 6.2.3 that we can compute a modal formula \( \chi \) such that for all finite transitive trees \( \mathcal{T}^+ \) and all nodes \( u \) in \( \mathcal{T}^+ \),

\[
\mathcal{T}^+, u \vDash \psi \iff \mathcal{T}^+, u \vDash \chi.
\]

Given the connection between CoreXPath(\( \downarrow^+ \)) and modal logic (see Section 6.1.3), we can compute a node expression \( \xi \) of CoreXPath(\( \downarrow^+ \)) such that for all finite trees \( \mathcal{T} = (T, R, V) \) and all nodes \( u \) in \( \mathcal{T} \), we have

\[
\mathcal{T}^+, u \vDash \chi \iff u \text{ belongs to } \llbracket \xi \rrbracket_{\mathcal{T}},
\]

where \( \mathcal{T}^+ = (T, R^+, V) \). Putting everything together, we obtain that for all finite trees \( \mathcal{T} \) and all nodes \( u \) in \( \mathcal{T} \), \( \mathcal{T}, u \vDash \varphi(x) \) iff \( u \) belongs to \( \llbracket \xi \rrbracket_{\mathcal{T}} \). This finishes the proof that an MSO formula \( \varphi(x) \) is equivalent to a CoreXPath(\( \downarrow^+ \)) node expression iff \( \varphi(x) \) is invariant under transitive bisimulation on finite trees.

Now it is easy to see that the fact that \( \xi \) is computable from \( \varphi(x) \) does not depend on the fact that \( \varphi(x) \) was invariant under transitive bisimulation on finite trees. The second statement of the proposition immediately follows.

\[\square\]

To prove Theorem 6.2.1, it remains to show that (ii) and (iii) are equivalent, by exploiting the tight link between transitive bisimulations and the operation \( \text{COPY}_{u \rightarrow v}(\mathcal{T}) \).

The hardest direction is to show that (iii) implies (ii). Given a tree \( \mathcal{T} \) and its copy \( \text{COPY}_{u \rightarrow v}(\mathcal{T}) \), there is an obvious transitive bisimulation linking the two.
models. We call such a transitive bisimulation, a $\sim$-transitive bisimulation. Now the idea is to show that each transitive bisimulation $B$, can be represented as the composition of $\sim$-transitive bisimulations. It will then be easy that we can derive (ii) from (iii). In order to make these intuitions more precise, we introduce the following terminology.

The relation $\sim$ and its associated bisimulation Let $\mathcal{T} = (T, R, V)$ and $\mathcal{S} = (S, Q, U)$ be finite trees. We write $\mathcal{T} \Rightarrow \mathcal{S}$ if there are nodes $u$ and $v$ of $\mathcal{T}$ such that $v$ is a descendant of $u$ and $\mathcal{S}$ is isomorphic to $\text{COPY}_{u \rightarrow v}(\mathcal{T})$. We use the notation $\mathcal{T} \sim \mathcal{S}$ if $\mathcal{T} \Rightarrow \mathcal{S}$ or $\mathcal{S} \Rightarrow \mathcal{T}$.

We say that a relation $B \subseteq T \times S$ is a $\sim$-transitive bisimulation for $\mathcal{T}$ and $\mathcal{S}$ if one of the two following conditions holds. Either there exist $u$ and $v$ in $\mathcal{T}$ such that $\mathcal{S}$ is isomorphic to $\text{COPY}_{u \rightarrow v}(\mathcal{T})$ and $B$ is the relation

$$\{(w, w) \mid w \in \mathcal{T}\} \cup \{(w, c(w)) \mid (v, w) \in R^+\}.$$ 

Or there exist $u$ and $v$ in $\mathcal{S}$ such that $\mathcal{T}$ is isomorphic to $\text{COPY}_{u \rightarrow v}(\mathcal{S})$ and $B$ is the relation

$$\{(w, w) \mid w \in \mathcal{S}\} \cup \{(c(w), w) \mid (v, w) \in Q^+\}.$$ 

If $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is a sequence of finite trees such that $\mathcal{T}_i \sim \mathcal{T}_{i+1}$, for all $i \in \{1, \ldots, n-1\}$, we say that $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is a $\sim$-sequence between $\mathcal{T}_1$ and $\mathcal{T}_n$. A relation $B \subseteq \mathcal{T}_i \times \mathcal{T}_n$ is a $\sim$-transitive bisimulation for $\mathcal{T}_1, \ldots, \mathcal{T}_n$ if either $n = 1$ and $B$ is the identity or for all $i \in \{1, \ldots, n-1\}$, there exists a relation $B_i$ which is a $\sim$-bisimulation for $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ and $B = B_1 \circ \cdots \circ B_{n-1}$. A relation $B$ between two trees $\mathcal{T}$ and $\mathcal{S}$ is a $\sim$-transitive bisimulation if there is a $\sim$-sequence $\mathcal{T}_1, \ldots, \mathcal{T}_n$ between $\mathcal{T}$ and $\mathcal{S}$ such that $B$ is a $\sim$-transitive bisimulation for $\mathcal{T}_1, \ldots, \mathcal{T}_n$.

6.2.5. Lemma. Let $B$ be a total transitive bisimulation between two finite trees $\mathcal{T}$ and $\mathcal{S}$. Then there exists a $\sim$-transitive bisimulation between $\mathcal{T}$ and $\mathcal{S}$ that is included in $B$.

Proof The proof is by induction on the depth of $\mathcal{T}$. If the depth of $\mathcal{T}$ is 1, then $\mathcal{T}$ and $\mathcal{S}$ are isomorphic and the lemma trivially holds.

For the induction step, suppose that $\mathcal{T}$ has depth $n + 1$. Let $r_0$ be the root of $\mathcal{T}$ and $s_0$ the root of $\mathcal{S}$. Let also $u_0, \ldots, u_k$ be the children of $r_0$ and $v_0, \ldots, v_m$ the children of $s_0$. Note that since $\mathcal{T}$ and $\mathcal{S}$ are linked by a total bisimulation, the labels of $r_0$ and $s_0$ are the same and the depth of $\mathcal{S}$ is $n + 1$.

First, we define $\mathcal{Q}$ as the tree obtained by taking the disjoint union of the trees $\{\mathcal{T}_u \mid u \text{ child of } r_0\}$ and $\{\mathcal{S}_v \mid v \text{ child of } s_0\}$ and by adding a root $r$ to it, the label of which is the label of $r_0$. We show that there exist a $\sim$-sequence between $\mathcal{T}$ and $\mathcal{Q}$ and a $\sim$-transitive bisimulation for this sequence that is a subset of $\{(r_0, r)\} \cup \{(u, u) \mid u \in \mathcal{T}\} \cup B$. 
For each child $v$ of $s_0$, there exists a node $f(v)$ in $\mathcal{T}$ such that $f(v)$ and $v$ are linked by $B$. It follows that $B \cap (\mathcal{T}_{f(v)} \times S_v)$ is a total transitive bisimulation between $\mathcal{T}_{f(v)}$ and $S_v$. By induction hypothesis, there exist a ~-sequence of finite trees between $\mathcal{T}_{f(v)}$ and $S_v$ and a ~-transitive bisimulation $B^v$ associated with this sequence and included in $B$.

Next, let $\mathcal{T}'$ be the tree obtained by taking the disjoint union of $\{\mathcal{T}_u \mid u \text{ child of } r_0\}$ and copies of the finite trees $\{\mathcal{T}_{f(v)} \mid v \text{ child of } s_0\}$ and by adding a root to it, the label of which is the label of $r_0$. If $w$ belongs to a tree $\mathcal{T}_{f(v)}$ (where $v$ is a child of $v_0$), we denote by $c_v(w)$ the copy of $w$ that belongs to the copy of $\mathcal{T}_{f(v)}$ in $\mathcal{T}'$. By definitions of ~ and $\mathcal{T}'$, it is easy to see that there exists a ~-sequence $\mathcal{T}_0, \ldots, \mathcal{T}_m$ between $\mathcal{T}$ and $\mathcal{T}'$. Moreover, the relation $B$ given by:

$$B = \{(w, w) \mid w \in \mathcal{T}\} \cup \{(w, c_v(w)) \mid v \text{ child of } s_0, w \text{ descendant of } f(v) \text{ or } w = f(v)\},$$

is a ~-transitive bisimulation for this sequence.

Now we define a ~-sequence of trees $\mathcal{T}_m, \ldots, \mathcal{T}_{2m}$ and ~-transitive bisimulations $B_m, \ldots, B_{2m}$ such that $\mathcal{T}_{2m} = Q$ and for all $m + 1 \leq i \leq 2m$, $B_i$ is a ~-transitive bisimulation for a ~-sequence between $\mathcal{T}_{i-1}$ and $\mathcal{T}_i$ and $B_m \circ \cdots \circ B_{m+i}$ is a subset of

$$\{(w, w) \mid w \in \mathcal{T}\} \cup \{(c_v(w), c_v(w)) \mid j > i, w \in \mathcal{T}_{f(v)}\} \cup \{(c_v(w), t) \mid j \leq i, (w, t) \in B^v\}.$$

The definitions of $\mathcal{T}_m, \ldots, \mathcal{T}_{2m}$ and $B_m, \ldots, B_{2m}$ are by induction. The tree $\mathcal{T}_m$ has been previously defined and is equal to $\mathcal{T}'$. The relation $B_m$ is defined as $\{(w, w) \mid w \in \mathcal{T}_m\}$. For the induction step, take $0 \leq i \leq m - 1$. We define $\mathcal{T}_{m+i+1}$ as the tree obtained by replacing in $\mathcal{T}_{m+i}$, the subtree with root $c_{v_{i+1}}(f(v_{i+1}))$ by the tree $S_{v_{i+1}}$. That is, we replace the copy of $\mathcal{T}_{f(v_{i+1})}$ by the tree $S_{v_{i+1}}$. Since there is a ~-sequence between $\mathcal{T}_{f(v_{i+1})}$ and $S_{v_{i+1}}$, we can easily construct from it a ~-sequence between the trees $\mathcal{T}_{m+i}$ and $\mathcal{T}_{m+i+1}$. Moreover, we may assume that

Figure 6.2: From $\mathcal{T}$ to $\mathcal{T}'$. 
Chapter 6. CoreXPath restricted to the descendant relation

Figure 6.3: From $T'$ to $Q$.

there is a $\sim$-transitive bisimulation $B_{m+i+1}$ associated to this sequence, such that

$B_{m+i+1} = \{(w, w) \mid w \neq f(v_{i+1}) \text{ and } w \text{ is not a descendant of } c_{v_{i+1}}(f(v_{i+1}))\}$

$\cup \{(c_{v_{i+1}}(w), t) \mid (w, t) \in B_{v_{i+1}}\}.$

It is routine to check that $T_{m+i+1}$ and $B_{m+i+1}$ satisfy the required properties.

Putting everything together, we obtain a sequence $T_0, \ldots, T_{2m}$ such that $T_0 = T$, $T_{2m} = Q$ and for all $0 \leq i < 2m$, there is a $\sim$-sequence between $T_i$ and $T_{i+1}$. Moreover, the $\sim$-transitive bisimulation $B \circ B_1 \circ \cdots \circ B_{2m}$ between $T_0$ and $T_{2m}$ is equal to

$\{(w, w) \mid w \in T\} \cup \{(w, t) \mid (w, t) \in B\},$ for some child $v$ of $v_0$.

It follows from the fact that $B^o \subseteq B$ for all children $v$ of $v_0$, that $B_1 \circ \cdots \circ B_{2m}$ is a subset of $\{(r_0, r)\} \cup \{(u, u) \mid u \in T\} \cup B$. This finishes the proof that there exist a $\sim$-sequence between $T$ and $Q$ and a $\sim$-transitive bisimulation for this sequence that is a subset of $\{(r_0, r)\} \cup \{(u, u) \mid u \in T\} \cup B$.

Similarly we also obtain a $\sim$-sequence $S_1, \ldots, S_l$ between $Q$ and $S$ and a $\sim$-transitive bisimulation for this sequence that is a subset of $\{(r_0, r)\} \cup \{(v, v) \mid v \in S\} \cup B$. We can deduce that the sequence of finite trees $T_1, \ldots, T_n, S_2, \ldots, S_l$ is a $\sim$-sequence between $T$ and $S$ and there is a $\sim$-transitive bisimulation associated with this sequence and included in $B$. \qed

6.2.6. PROPOSITION. An MSO formula $\phi(x)$ is invariant under transitive bisimulation on finite trees iff $\phi(x)$ is invariant under the subtree and copy operations.

Proof The direction from left to right follows easily from the facts that $\phi(x)$ is invariant under transitive bisimulation on finite trees and that the relation $\{(w, w) \mid w \in T\} \cup \{(w, c(w)) \mid vR^+ w\}$ is a transitive bisimulation between $T$ and $\text{COPY}_{u \rightarrow v}(T)$.

For the direction from left to right, suppose that an MSO formula $\phi(x)$ is invariant under the subtree and copy operations. We have to prove that $\phi(x)$ is invariant under transitive bisimulation on finite trees. Let $B$ be a transitive bisimulation between two finite trees $T$ and $T'$ and suppose that $(u, u')$ belongs
6.3. CoreXPath(↓↑) path expressions

to B. Then, \( B \cap (T_u \times T'_u) \) is a total transitive bisimulation between \( T_u \) and \( T'_u \). It follows from Lemma 6.2.5 that there exists a \( \sim \)-sequence between \( T_u \) and \( T'_u \). Using the fact that is invariant under the copy operation, we can check by induction on the length of the \( \sim \)-sequence that \( T_u, u \models \varphi(x) \) iff \( T'_u, u' \models \varphi(x) \). Putting this together with the fact that is invariant under the subtree operation, we obtain that \( T, u \models \varphi(x) \) iff \( T', u' \models \varphi(x) \).

Using Proposition 6.2.4 and Proposition 6.2.6, we obtain Theorem 6.2.1. Putting the second statement of Theorem 6.2.1 together with the decidability of MSO on finite trees, we obtain the following result.

6.2.7. Corollary. It is decidable whether an MSO formula is equivalent to a CoreXPath(↓↑) node expression.

6.2.8. Remark. We would like to mention that although the equivalence between (i) and (iii) is specific to the setting of finite trees, the equivalence between (i) and (ii) can be adapted the case of trees. Note that strictly speaking, a CoreXPath(↓↑) node expression cannot be, by definition, evaluated on an infinite tree. However, there is an obvious way to extend the interpretation of CoreXPath(↓↑) node expressions to the setting of infinite trees. Using a similar proof to the one we gave for Theorem 6.2.1, we can show that an MSO formula \( \varphi(x) \) is equivalent to CoreXPath(↓↑) node expression on trees iff \( \varphi(x) \) is invariant under transitive bisimulation.

6.3 CoreXPath(↓↑) path expressions

Now we will adapt this method in order to obtain a similar characterization for path expressions. As before, the characterization is twofold. We provide a first characterization that is formulated in terms of transitive bisimulations. We also give another characterization based on the link between CoreXPath(↓↑) path expressions and the operation COPY. We start by introducing some terminology.

**Equivalence** An MSO formula \( \varphi(x, y) \) is equivalent to a CoreXPath(↓↑) path expression \( A \) on finite trees if for all finite trees \( T \) and for all nodes \( u, v \) in \( T \), \( T, (u, v) \models \varphi(x, y) \) iff \( (u, v) \) belongs to \( \llbracket A \rrbracket_T \). When this happens, we write \( \varphi(x, y) \equiv A \).

In the setting of programs, the notion corresponding to invariance under bisimulation, is the notion of safety for bisimulations, which was introduced by Johan van Benthem [Ben98].

**Safety for bisimulations** An MSO formula \( \varphi(x, y) \) is safe for (transitive) bisimulations on finite trees if for all finite trees \( T, T' \), (transitive) bisimulations \( B \subseteq T \times T' \), pairs \( (u, u') \in B \), and nodes \( v \in T \), if \( T, (u, v) \models \varphi(x, y) \), then there exists a node \( v' \in T' \) such that \( (v, v') \in B \) and \( T', (u', v') \models \varphi(x, y) \).
Chapter 6. CoreXPath restricted to the descendant relation

Invariance under the subtree and the copy operations Let $\varphi(x, y)$ be an MSO formula. We say that $\varphi(x, y)$ is invariant under the subtree operation (on finite trees) if for all finite trees $\mathcal{T}$ and for all nodes $u, v$ in $\mathcal{T}$ such that $v$ is a descendant of $u$,

$$\mathcal{T}, (u, v) \models \varphi(x, y) \iff \mathcal{T}_u, (u, v) \models \varphi(x, y).$$

The formula $\varphi(x, y)$ is invariant under the copy operation (on finite trees) if for all finite trees $\mathcal{T}$, for all nodes $u, v, w, t$ in $\mathcal{T}$ such that $v$ is a descendant of $u$ and $t$ a descendant of $v$, we have

$$\mathcal{T}, (w, t) \models \varphi(x, y) \iff \text{COPY}_{u \rightarrow v}(\mathcal{T}), (w, t) \models \varphi(x, y), \quad (6.3)$$

$$\text{COPY}_{u \rightarrow v}(\mathcal{T}), (w, c(t)) \models \varphi(x, y) \quad \text{implies} \quad \mathcal{T}, (w, t) \models \varphi(x, y). \quad (6.4)$$

6.3.1. Theorem. Let $\varphi(x, y)$ be an MSO formula. The following are equivalent:

(i) $\varphi(x, y)$ is equivalent to a CoreXPath($\downarrow^+$) path expression,

(ii) $\varphi(x, y)$ is safe for transitive bisimulations on finite trees,

(iii) it is the case that $\varphi(x, y)$ is invariant under the subtree and copy operations.

Moreover, given an MSO formula $\varphi(x, y)$, we can compute a CoreXPath($\downarrow^+$) path expression $A$ such that $\varphi(x, y)$ is equivalent to a CoreXPath($\downarrow^+$) path expression iff $\varphi(x, y) \equiv A$. 
The structure of the proof is the same as the one for node expressions. It is based on versions of the Janin-Walukiewicz theorem and the de Jongh fixpoint theorem, adapted to the setting of $\mu$-programs (instead of $\mu$-formulas). We recall the syntax for the $\mu$-programs. The $\mu$-programs are given by

$$\theta ::= R | \varphi? | \theta \cup \theta | \theta^*,$$

where $\varphi$ is a $\mu$-sentence.

Given a Kripke model, these $\mu$-programs are interpreted as binary relations over the model. These binary relations are defined by induction on the complexity of the programs. We only recall that the interpretation of $\theta^*$ is the reflexive transitive closure of the relation corresponding to $\theta$. For more details, see Section 2.6.

There exists an expressivity result for $\mu$-programs, which is the equivalent of the Janin-Walukiewicz theorem. It was proved by Marco Hollenberg [Hol98b] (for more details about this result, see Section 5.5 in Chapter 5). We show here how to derive this result from Theorem 5.5.3, in the special case where we restrict the class of structures to finite trees. Recall that an MSO formula $\varphi(x, y)$ is equivalent on finite trees to a $\mu$-program $\theta$ if for all finite trees $T$ and nodes $u, v$ in $T$, $T, (u, v) \models \varphi(x, y)$ iff $(u, v) \in \llbracket \theta \rrbracket_T$.

**6.3.2. Theorem.** [from [Hol98b]] An MSO formula $\varphi(x, y)$ is safe for bisimulations on finite trees iff it is equivalent on finite trees to a $\mu$-program.

Moreover, given an MSO formula $\varphi(x, y)$, we can compute a $\mu$-program $\theta$ such that $\varphi(x, y)$ is equivalent on finite trees to a $\mu$-program iff $\varphi(x, y)$ is equivalent to $\theta$ on finite trees.

**Proof** The proof mainly relies on a variant of Theorem 5.5.3. The method used to derive the result from Theorem 5.5.3 is the one used by Johan van Benthem in [Ben98].

It is easy to show by induction on the complexity of $\mu$-programs (and using the fact that $\mu$-formulas are invariant under bisimulation) that a $\mu$-program is safe for bisimulations. Hence, it is sufficient to show that given an MSO formula $\varphi(x, y)$, we can compute a $\mu$-program $\theta$ such that if $\varphi(x, y)$ is safe for bisimulations on finite trees, then $\varphi(x, y)$ and $\theta$ are equivalent on finite trees.

Let $\varphi(x, y)$ be an MSO formula. Let $\psi(x)$ be the MSO formula $\exists y(\varphi(x, y) \land P(y))$, where $P$ is a unary predicate corresponding to a fresh proposition letter $p$ (i.e. $p$ does not occur in $\varphi(x, y)$). By Theorem 6.2.2, we can compute a $\mu$-sentence $\chi$ such that $\psi(x)$ is invariant under bisimulation on finite trees iff $\psi(x)$ and $\chi$ are equivalent on finite trees. It is easy to see that if $\varphi(x, y)$ is safe for bisimulations, then $\psi(x)$ is invariant under bisimulation on finite trees and in particular, $\psi(x)$ is equivalent to $\chi$ on finite trees.

Now the formula $\chi$ is completely additive with respect to $p$ on finite trees. That is, for all finite trees $T = (T, R, V)$ and all nodes $u$ in $T$,

$$T, u \models \psi \text{ iff there is a node } v \in V(p) \text{ such that } T[p \mapsto \{v\}], u \models \psi.$$
It follows from Theorem 5.5.3 that for each formula that is completely additive with respect to \( p \), we can compute an equivalent formula in the syntactic fragment \( \mu\text{ML}_A(p) \). The proof of Theorem 5.5.3 could be easily adapted to show that for each formula \( \chi \) that is completely additive in \( p \) on finite trees, we can compute a formula in \( \mu\text{ML}_A(p) \), that is equivalent to \( \chi \) on finite trees.

Moreover, as observed in Section 5.5 of Chapter 5, a formula belongs to \( \mu\text{ML}_S(p) \) iff it is equivalent to a formula of the form \( \langle \theta \rangle p \), where \( p \) does not occur in the program \( \theta \). In fact, given a formula in \( \mu\text{ML}_S(p) \), we can compute a \( \mu \)-program \( \theta \) such that the formula is equivalent to \( \langle \theta \rangle p \). Putting everything together, we can compute a \( \mu \)-program \( \theta \) in which \( p \) does not occur, such that that \( \chi \) is equivalent on finite trees to \( \langle \theta \rangle p \).

Recall that if \( \phi(x, y) \) is safe for bisimulations, then \( \psi(x) \) is equivalent to \( \chi \) on finite trees. Hence, if \( \phi(x, y) \) is safe for bisimulations, \( \psi(x) \) is equivalent to \( \langle \theta \rangle p \) on finite trees. It follows from the definition of \( \psi(x) \) that if \( \psi(x) \) is equivalent to \( \langle \theta \rangle p \) on finite trees, then \( \phi(x, y) \) is equivalent to \( \theta \) on finite trees. This finishes the proof that given an MSO formula \( \phi(x, y) \), we can compute a \( \mu \)-program \( \theta \) such that if \( \phi(x, y) \) is safe for bisimulations on finite trees, then \( \phi(x, y) \) and \( \theta \) are equivalent on finite trees.

We can also prove a variant of Theorem 6.2.3, which applies to programs. Recall that a modal program is a \( \mu \)-program which does not contain any Kleene star \( ^* \) and all its subprograms of the form \( \phi? \) are such that \( \phi \) is modal (a precise definition was given in Section 6.1).

6.3.3. THEOREM. For all \( \mu \)-programs \( \theta \), we can compute a modal program \( \lambda \) such that for all finite transitive trees \( T^+ \), we have \( \llbracket \theta \rrbracket_{T^+} = \llbracket \lambda \rrbracket_{T^+} \).

Proof The proof consists in showing that on finite transitive trees, each \( \mu \)-program is equivalent to a finite disjunction of modal programs which are in a special shape. We call these modal programs basic. More precisely, we say that a modal program is a basic modal program if it belongs to the language defined by the following grammar

\[
\theta \ ::= \ \phi? \mid (\theta; R; \phi?),
\]

where \( \phi \) is a modal formula. We show that each \( \mu \)-program is equivalent to a finite disjunction of basic modal programs on finite transitive trees. The proof is by induction on the complexity of the \( \mu \)-programs.

First, assume that \( \theta \) a program of the form \( \phi? \) (where \( \phi \) is a \( \mu \)-sentence). Then it follows from Theorem 6.2.3 that \( \phi \) is equivalent to a modal formula \( \psi \) on finite transitive trees. Therefore, \( \theta \) is equivalent to the basic modal program \( \psi? \) on finite transitive trees.

The cases where \( \theta \) is the program \( R \) or a program of the form \( \theta_1 \cup \theta_2 \), are immediate. Next assume that \( \theta \) is of the form \( \theta_1; \theta_2 \). By induction hypothesis,
there exist sets $\Gamma_1$ and $\Gamma_2$ of basic modal programs such that $\theta_1 = \lor \Gamma_1$ and $\theta_2 = \lor \Gamma_2$. It follows that $\theta$ is equivalent to the modal program $\lor \{\gamma_1; \gamma_2 \mid \gamma_1 \in \Gamma_1, \gamma_2 \in \Gamma_2\}$. It remains to check that for all $\gamma_1$ in $\Gamma_1$ and all $\gamma_2$ in $\Gamma_2$, $\gamma_1; \gamma_2$ is equivalent to a basic modal program. Since $\gamma_1$ and $\gamma_2$ are basic programs, there are modal formulas $\varphi_1, \ldots, \varphi_n$ and $\psi_1, \ldots, \psi_k$ such that $\gamma_1 = \varphi_1?; R; \ldots; R; \varphi_n?$ and $\gamma_2 = \psi_1?; R; \ldots; R; \psi_k?$. Therefore, $\gamma_1; \gamma_2$ is equivalent to the basic program

$$\varphi_1?; R; \ldots; R; (\varphi_n \land \psi_1)?; R; \psi_2; R; \ldots; R; \psi_k?.$$  

The only case left is where $\theta$ is a $\mu$-program of the form $\lambda^\ast$. By induction hypothesis, $\lambda$ is equivalent to the modal program $\lor \Gamma$, for some set $\Gamma$ of basic modal programs. We prove that $\theta$ is equivalent to the disjunction of the set $\Delta$ given by:

$$\Delta = \{\gamma_1; \ldots; \gamma_n \mid n \in \mathbb{N}, \gamma_1, \ldots, \gamma_n \in \Gamma \text{ and } \gamma_1, \ldots, \gamma_n \text{ are pairwise distinct}\}.$$  

First, it is routine to check that $[\lor \Delta]_{\mathcal{T}^+}$ is a subset of $[\lambda^\ast]_{\mathcal{T}^+}$, on all finite transitive trees $\mathcal{T}^+$. So it remains to show that if $\mathcal{T}^+$ is a finite transitive tree and if the pair $(u, v)$ belongs to $[\lambda^\ast]_{\mathcal{T}^+}$, then there is a program $\delta$ in $\Delta$ such that $(u, v)$ belongs to $[\delta]_{\mathcal{T}^+}$.

Let $(u, v)$ be such a pair. Since $\theta$ is equivalent to $(\lor \Gamma)^\ast$ on finite transitive trees, there are programs $\gamma_1, \ldots, \gamma_n$ in $\Gamma$ such that $(u, v)$ belongs to $[\gamma_1; \ldots; \gamma_n]$. The problem is that $\gamma_1, \ldots, \gamma_n$ might not be pairwise distinct. First, suppose that for all $i \in \{1, \ldots, n\}$, the program $\gamma_i$ is of the form $\varphi_i$? (for some modal formula $\varphi_i$). Then, it is easy to see that if there are $i$ and $j$ such that $\varphi_i = \varphi_j$, we can delete the program $\gamma_j$ from the list $\gamma_1, \ldots, \gamma_n$, without modifying the fact that $(u, v)$ belongs to $[\gamma_1; \ldots; \gamma_n]$.

Next suppose that there is at least one program $\gamma_{i_0}$ which is not of the form $\varphi$? (for some modal formula $\varphi$). First, we may assume that in fact no program $\gamma_i$ is of the form $\psi$?. If there were such a program, say $\gamma_i = \psi$?, then we could remove $\gamma_i$ from the list $\gamma_1, \ldots, \gamma_n$ and it will still be the case that $(u, v)$ belongs to the relation associated to the program $\gamma_1; \ldots; \gamma_n$.

Now suppose that there are $i$ and $j$ such that $i < j$ and $\gamma_i = \gamma_j$. Since $\gamma_i$ is not a program of the form $\psi$?, it follows from the induction hypothesis that there exists a formula $\psi$ and a basic modal program $\gamma$ such that $\gamma_i = \psi$?; $R$; $\gamma$. Since $(u, v)$ belongs to $[\gamma_1; \ldots; \gamma_n]$ and $\gamma_i = \gamma_j = \psi$?; $R$; $\gamma$, there are nodes $u_1$ and $u_2$ such that

- $(u, u_1)$ belongs to $[\gamma_1; \ldots; \gamma_{i-1}; \varphi]$,
- $(u_1, u_2)$ belongs to $[R; \gamma; \gamma_{i+1}; \ldots; \gamma_{j-1}; \varphi]; R]$,
- $(u_2, v)$ belongs to $[\gamma; \gamma_{j+1}; \ldots; \gamma_n]$.

Note that if $(u_1, u_2)$ belongs to $[R; \gamma; \gamma_{i+1}; \ldots; \gamma_{j-1}; \varphi]; R]$, then in particular, $u_2$ is a descendant of $u_2$. That is, $(u_1, u_2)$ belongs to $[R]$. So we obtain that
• \((u, u_1)\) belongs to \([\gamma_1; \ldots; \gamma_{i-1}; \varphi]\),

• \((u_1, u_2)\) belongs to \([R]\),

• \((u_2, v)\) belongs to \([\gamma; \gamma_{j+1}; \ldots; \gamma_n]\).

That is, \((u, v)\) belongs to \([\gamma_1; \ldots; \gamma_i; \gamma_{j+1}; \ldots; \gamma_n]\). This means that whenever there are \(i\) and \(j\) such that \(i < j\) and \(\gamma_i = \gamma_j\), we can remove the programs \(\gamma_{i+1}, \ldots, \gamma_j\) from the list \(\gamma_1, \ldots, \gamma_j\) and it is still the case that \((u, v)\) belongs to the relation associated to the program \(\gamma_1; \ldots; \gamma_n\). By repeating this operation, we may assume that \((u, v)\) belongs to \([\gamma_1; \ldots; \gamma_n]\), where the programs \(\gamma_1, \ldots, \gamma_n\) are pairwise distinct. Therefore, \((u, v)\) belongs to \([\bigvee \Delta]\) and this finishes the proof that for all \(\mu\)-programs \(\theta\), there exists a modal program \(\lambda\) which is equivalent to \(\theta\) on finite transitive trees. A careful inspection of the proof shows that \(\lambda\) can be effectively computed from \(\theta\).

We can now prove that CoreXPath(\(\downarrow^+\)) path expressions correspond to the MSO(\(\downarrow\)) formulas that are safe for bisimulations.

**6.3.4. Proposition.** An MSO formula \(\varphi(x, y)\) is equivalent on finite trees to a CoreXPath(\(\downarrow^+\)) path expression iff \(\varphi(x, y)\) is safe for transitive bisimulations on finite trees.

Moreover, given an MSO formula \(\varphi(x, y)\), we can compute a CoreXPath(\(\downarrow^+\)) path expression \(A\) such that \(\varphi(x, y)\) is equivalent to a CoreXPath(\(\downarrow^+\)) path expression iff \(\varphi(x, y) \equiv A\).

**Proof** First, we show that an MSO formula \(\varphi(x, y)\) is equivalent on finite trees to a CoreXPath(\(\downarrow^+\)) path expression iff \(\varphi(x, y)\) is safe for transitive bisimulations on finite trees. For the direction from left to right, the proof is a standard induction on the complexity of CoreXPath(\(\downarrow^+\)) path expressions.

For the direction from right to left, let \(\varphi(x, y)\) be an MSO formula that is safe for transitive bisimulations on finite trees. By Theorem 6.3.2, we can compute a \(\mu\)-program \(\theta\) such that \(\varphi(x, y)\) and \(\theta\) are equivalent on finite trees iff \(\varphi(x, y)\) is safe for bisimulations on finite trees. Since \(\varphi(x, y)\) is safe for transitive bisimulations on finite trees, \(\varphi(x, y)\) is safe for bisimulations on finite trees. Hence, \(\varphi(x, y)\) and \(\theta\) are equivalent on finite trees.

Now we show that for all finite trees \(T = (T, R, V)\), we have that \([\theta]_T\) is equal to \([\theta]_{T^+}\). Take a finite tree \(T = (T, R, V)\) and nodes \(u, v\) in \(T\). We have to show that

\[(u, v) \in [\theta]_T \iff (u, v) \in [\theta]_{T^+}.

(6.5)

Let \(T^+\) be the transitive tree \((T, R^+, V)\) and let \(S\) be the unraveling of the pointed model \((T^+, u)\) (see Section 2.6). Recall that \(S\) is a finite tree the root of which is \(u\). Its domain is the set of paths of \(T^+\) with starting point \(u\).
6.3. CoreXPath($\downarrow^+$) path expressions

For the direction from left to right of (6.5), suppose that $(u, v)$ belongs to $[[\theta]]_T$. The canonical bisimulation $B$ between $S$ and $T^+$ is a transitive bisimulation between $S$ and $T$ such that $(u, u)$ belongs to $B$. Since $\theta$ and $\varphi(x, y)$ are equivalent on finite trees and $\varphi(x, y)$ is safe for transitive bisimulations on finite trees, it follows from $(u, v) \in [[\theta]]_T$ and $(u, u) \in B$ that for some $s = (u_i)_{i \leq n}$ in $S$, we have $(v, s) \in B$ and $(u, s) \in [[\theta]]_S$. By definition of $B$, $(v, s) \in B$ implies that $u_n = v$. Now as $\theta$ is a $\mu$-program, it is safe for bisimulations. Since $(u, s) \in [[\theta]]_S$ and the pair $(u, u)$ belongs to the canonical bisimulation $B$ between $S$ and $T^+$, we have $(u, v') \in [[\theta]]_{T^+}$ and $(v', s) \in B^c$, for some $v'$ in $T^+$. By definition of the canonical bisimulation $B$, this can only be the case if $u_n = v'$. Since $u_n = v$, it follows that $v' = v$. Putting everything together, we obtain that $(u, v)$ belongs to $[[\theta]]_{T^+}$.

For the direction from right to left of (6.5), suppose that $(u, v)$ belongs to $[[\theta]]_{T^+}$. Since $(u, u)$ belongs to the bisimulation $B$ and $\alpha$ is safe for bisimulation, there exists $s = (u_i)_{i \leq n}$ such that $(u, s) \in [[\theta]]_S$ and $(v, s) \in B$. By definition of $B$, this can only happen if $u_n = v$. Now we have that $(u, s) \in [[\theta]]_S$, $(u, u)$ belongs to the transitive bisimulation $B$ and $\alpha$ is equivalent to $\varphi(x, y)$, which is safe for transitive bisimulations. Therefore, there exists $v' \in T$ such that $(u, v') \in [[\theta]]_T$ and $(s, v') \in B$. Since $(s, v')$ belongs to $B$, we have $u_n = v'$. We established earlier that $u_n = v$. Hence, $v = v'$. Putting everything together, we have $(u, v) \in [[\theta]]_T$ and this finishes the proof that $[[\theta]]_T = [[\theta]]_{T^+}$.

It also follows from Theorem 6.3.3 that $\theta$ is equivalent on finite transitive trees to a modal program $\lambda$. Given the connection between CoreXPath($\downarrow^+$) and modal logic (see Section 6.1.3), there exists a CoreXPath($\downarrow^+$) path expression $A$ such that for all finite trees $T = (T, R, V)$ and for all nodes $u, v \in T$.

$$(u, v) \in [[\lambda]]_{T^+} \iff [A]_T,$$

where $T^+ = (T, R^+, V)$. Putting everything together, we found a CoreXPath($\downarrow^+$) path expression $A$ such that for all finite trees $T$ and all nodes $u, v$ in $T$, $T, (u, v) \vdash \varphi(x, y)$ iff $(u, v)$ belongs to $[[A]]_T$. This finishes the proof of the first statement.

Now it is easy to see that the fact that $A$ is computable from $\varphi(x, y)$ does not depend on the fact that $\varphi(x, y)$ was safe for transitive bisimulations on finite trees. The second statement of the proposition immediately follows.

\[ \square \]

6.3.5. Proposition. An MSO formula $\varphi(x, y)$ is safe for transitive bisimulations on finite trees iff $\varphi(x, y)$ is invariant under the subtree and copy operations.

**Proof** The direction from left to right follows easily from the facts that $\varphi(x, y)$ is safe for transitive bisimulations on finite trees and that the relation $\{(w, w) \mid w \in T\} \cup \{(w, c(w)) \mid vR^+w\}$ is a transitive bisimulation between $T$ and $\text{COPY}_{u \rightarrow v}(T)$. 

For the direction from right to left, suppose that $\varphi(x, y)$ is an MSO formula that is invariant under the subtree and copy operations. In order to show that $\varphi(x, y)$ is safe for transitive bisimulation on finite trees, let $B$ be a transitive
bisimulation between two finite trees \( \mathcal{T} \) and \( \mathcal{S} \). Suppose that \((u_0, v_0)\) belongs to \( B \) and that there is a node \( u_1 \) such that \( \mathcal{T}, (u_0, u_1) \models \varphi(x, y) \). We have to prove that there exists a node \( v_1 \in \mathcal{S} \) such that \((u_1, v_1)\) belongs to \( B \) and \( \mathcal{S}, (v_0, v_1) \models \varphi(x, y) \).

First observe that \( B \cap (\mathcal{T}_{u_0} \times \mathcal{S}_{v_0}) \) is a total transitive bisimulation between \( \mathcal{T}_{u_0} \) and \( \mathcal{S}_{v_0} \). By Lemma 6.2.5, there exist finite trees \( \mathcal{T}_1, \ldots, \mathcal{T}_n \), relations \( B_1, \ldots, B_{n_1} \) such that \( \mathcal{T}_i = \mathcal{T}_{u_0}, \mathcal{S}_i = \mathcal{S}_{v_0}, B_i \) is a \( \sim \)-transitive bisimulation between \( \mathcal{T}_i \) and \( \mathcal{T}_{i+1} \) (for all \( i \in \{1, \ldots, n - 1\} \)) and \( B_1 \circ \cdots \circ B_{n-1} \) is included in \( B \). We denote by \( B_0 \) the transitive bisimulation \( \{(w, w) \mid w \in \mathcal{T}_1\} \) between \( \mathcal{T}_1 \) and \( \mathcal{T}_i \).

Now we prove by induction on \( i \) that for all \( 1 \leq i \leq n \),

there exists \((u_i, w_i) \in B_0 \circ \cdots \circ B_{i-1} \) such that \( \mathcal{T}_i, (r_i, w_i) \models \varphi(x, y) \), \( (6.6) \)

where \( r_i \) is the root of \( \mathcal{T}_i \). The case where \( i = 1 \) is immediate as \( \mathcal{T}, (u_0, u_1) \models \varphi(x, y) \) and \( \mathcal{T}_1 = \mathcal{T}_{u_0} \). Let us turn to the induction step \( i + 1 \). By induction hypothesis, there exists \((u_i, w_i) \in B_0 \circ \cdots \circ B_{i-1} \) such that \( \mathcal{T}_i, (r_i, w_i) \models \varphi(x, y) \). Since \( B_i \) is a \( \sim \)-transitive bisimulation between \( \mathcal{T}_i \) and \( \mathcal{T}_{i+1} \), either \( \mathcal{T}_i \Rightarrow \mathcal{T}_{i+1} \) or \( \mathcal{T}_{i+1} \Rightarrow \mathcal{T}_i \). We suppose that \( \mathcal{T}_{i+1} \Rightarrow \mathcal{T}_i \). The other case is in fact easier. So there are nodes \( u, v \) of \( \mathcal{T}_{i+1} \) such that \( \mathcal{T}_i \) is equal to \( \text{COPY}_{u \rightarrow v}(\mathcal{T}_{i+1}) \). Since \( B_i \) is a \( \sim \)-transitive bisimulation and \((w_i, w_{i+1}) \in B_i \), either \( w_i \) belongs to \( \mathcal{T}_{i+1} \) or \( w_i = c(w) \) for some descendant \( w \) of \( v \) in \( \mathcal{T}_{i+1} \).

If \( w_i \) belongs to \( \mathcal{T}_{i+1} \), we can define \( w_{i+1} \) as \( w_i \) and by the fact that \( \varphi(x, t) \) is invariant under the copy operation (see condition (6.3)), it is the case that \( \mathcal{T}_{i+1}, (r_{i+1}, w_{i+1}) \models \varphi(x, y) \). If \( w_i = c(w) \) for some descendant \( w \) of \( v \) in \( \mathcal{T}_{i+1} \), we can define \( w_{i+1} \) as \( w \) and using the fact that \( \varphi(x, y) \) is invariant under the copy operation (see condition (6.4)), we have \( \mathcal{T}_{i+1}, (r_{i+1}, w_{i+1}) \models \varphi(x, y) \). This finishes the proof of \( (6.6) \).

Next let \( v_1 \) be a node such that \((u_1, v_1)\) belongs to \( B_1 \circ \cdots \circ B_{n-1} \) and \( \mathcal{T}_n, (r_n, v_1) \models \varphi(x, y) \). That is, \( \mathcal{S}_{v_0}, (v_0, v_1) \models \varphi(x, y) \). Using the fact that \( \varphi(x, y) \) is invariant under the subtree operation, we get that \( \mathcal{S}, (v_0, v_1) \models \varphi(x, y) \). Finally, since \( B_1 \circ \cdots \circ B_{n-1} \subseteq B \), \((u_1, v_1)\) belongs to \( B \). Putting everything together, we found a node \( v_1 \) in \( \mathcal{S} \) such that \((u_1, v_1)\) belongs to \( B \) and \( \mathcal{S}, (v_0, v_1) \models \varphi(x, y) \). □

Putting Proposition 6.3.5 and Proposition 6.3.4 together, we obtain a proof of Theorem 6.3.1. Putting the second statement of Theorem 6.3.1 together with the decidability of \( \text{MSO} \) on finite trees, we obtain the following result.

**6.3.6. COROLLARY.** It is decidable whether an \( \text{MSO} \) formula is equivalent to a CoreXPath(\( \downarrow^+ \)) path expression.

**6.3.7. REMARK.** Similarly to the observation at the end of the previous section, we can extend the equivalence between (i) and (ii) in Theorem 6.3.1 to the setting of infinite trees.
6.4 Conclusions

In this chapter, we gave a characterization of the MSO formulas that are equivalent to a CoreXPath($\downarrow^+$) node expression. First we formulated the characterization in terms of bisimulations, using classical results about fixpoints. We derived from this result another characterization which involves closure properties under some simple operations on finite trees. We gave a similar characterization for the MSO formulas that are equivalent to a CoreXPath($\downarrow^+$) path expression. From both characterizations, we could derive a decision procedure.

We could ask what is the complexity of the procedure (in Theorem 6.2.1) that, given an MSO formula $\varphi(x)$, determines whether $\varphi(x)$ is equivalent to a CoreXPath($\downarrow^+$) node expression. We did not look in details at this question but we suspect that this procedure may be non-elementary. A possible proof would be to reduce our problem to the satisfiability problem for MSO on finite trees, which is non-elementary. The same comment holds for the complexity of the procedure in Theorem 6.3.1.

It is an important open question whether there is a similar decidable characterization for full CoreXPath in terms of MSO. Thomas Place and Luc Segoufin [PS10] recently characterized the node expressions of an important fragment of CoreXPath, namely CoreXPath($\downarrow^+, \uparrow^+, \leftarrow^+, \rightarrow^+$).