Finding $\mathcal{EL}^+$ justifications using the Earley parsing algorithm

Riku Nortje[1,2] **Katarina Britz**[1,2] **Thomas Meyer**[1,2]

[1] Knowledge Systems Group, Meraka, CSIR,
PO Box 395, Pretoria 0001, South Africa

[2] School of Computing, University of South Africa,
PO Box 392, UNISA 0003, South Africa

Email: nortjeriku@gmail.com; {arina.britz;tommie.meyer}@meraka.org.za

Abstract

Module extraction plays an important role in the reuse of ontologies as well as in the simplification and optimization of some reasoning tasks such as finding justifications for entailments. In this paper we focus on the problem of extracting small modules for $\mathcal{EL}^+$ entailment based on reachability. We extend the current notion of (forward) reachability to obtain a bi-directional version, and show that the bi-directional reachability algorithm allows us to transform an $\mathcal{EL}^+$ ontology into a reachability preserving context free grammar (CFG). The well known Earley algorithm for parsing strings, given some CFG, is then applied to the problem of extracting minimal reachability-based axioms sets for subsumption entailments. We show that each reachability-based axiom set produced by the Earley algorithm corresponds to a possible Minimal Axiom Set (MinA) that preserves the given entailment. This approach has two advantages – it has the potential to reduce the number of subsumption tests performed during MinA extraction, as well to minimize the number of axioms for each such test.

1 Introduction

Reasoning tasks such as finding all justifications for an entailment are inherently hard, having at least an exponential worst case complexity. Even for description logics (DLs) such as $\mathcal{EL}^+$ (Suntisrivaraporn 2009) for which many reasoning tasks can be performed in polynomial time, the exponential nature of finding all justifications for an entailment is inescapable. Module extraction is one of the methods that aims to optimize the performance of this process by reducing the size of the ontology to a smaller subset of axioms that contains only the relevant axioms required for the entailment, thereby reducing the search space.

Extracting a minimal module is closely related to computing a deductive conservative extension of an ontology, which has been shown by Grau et al. (2007, 2008) to be intractable. They introduce syntactic locality-based modules, an approximation of minimal modules, that are more tractable and can be computed in polynomial time, whilst preserving all entailments.

Suntisrivaraporn (2009) introduced the notion of reachability-based modules for the DL $\mathcal{EL}^+$. Though the reachability-based module extraction algorithm differs from the syntactic locality-based algorithm, he proves that the modules extracted correspond to minimal syntactic locality-based modules.

The main criticism against reachability-based modules, as raised by Jianfeng Du & Ji (2009), is that, given an entailment $\mathcal{O} \models A \sqsubseteq B$, these methods extract a module for A and all concepts reachable from it without considering the super-concept B, resulting in large modules that in some cases do not reduce the size of the ontology at all. They propose a goal-directed algorithm for extracting just-preserving modules. The algorithm proceeds in two phases; the first, the off-line phase, transforms the ontology into a propositional program which preserves all logical relationships. The second phase, the on-line phase, utilizes the idea of maximally connected components, as used in SAT problem optimization, to extract a justification-preserving module. Experimental results show that modules obtained this way are smaller by an order of magnitude than their locality-based module counterparts.

Once a module has been extracted, various methods are used to find all justifications. A common approach is to systematically remove axioms that do not play a role in the entailment. After every iteration of the axiom removal procedure, a subsumption test determines if the entailment still holds in the resulting axiom set. This process continues until no more axioms can be removed. The resulting axiom set then constitutes a MinA. Subsumption testing is a computationally expensive procedure, even for $\mathcal{EL}^+$. Minimizing the number of subsumption tests during MinA extraction is therefore a primary concern when developing MinA extraction algorithms.

Our approach extends the reachability heuristic as introduced by Suntisrivaraporn (2009) to include backward reachability, thereby obtaining a bi-directional version of reachability. The heuristic allows us to make use of the well known Earley algorithm (Earley 1970) for parsing context free grammars (Jurafsky & Martin 2009) to compute all reachability-based paths between the sub- and super-concepts for an entailment. Each such path constitutes a minimal set of axioms such that reachability is preserved. Every reachability preserving axiom set does not guarantee entailment in itself, and therefore does not necessarily constitute a MinA. We focus on extracting all such minimal reachability preserving axiom sets. A standard subsumption test can then be employed in order to determine if the set constitutes a valid MinA. In this way we hope to reduce the number of subsumption tests performed during MinA extraction, as well as to minimize the number of axioms for each such test.

The Earley algorithm has been studied extensively in the literature and highly optimized software and hardware implementations exists (Chiang & Fu 1984, Pavlatos et al. 2003). At present we restrict our focus to $\mathcal{EL}^+$ which, because of its particular structure, allows us to transform any axiom into reachability preserving CFG production rules. The original Earley algorithm can then be used to extract all parse trees.

The rest of the paper is structured as follows. Section 2 contains the relevant background information

Name	Syntax	Semantics
top	$\top$	Δ^I
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^I \cap D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I : (x,y) \in r^I \wedge r \in C^I\}$
general concept inclusion (GCI)	$C \sqsubseteq D$	$C^I \subseteq D^I$
role inclusion (RI)	$r_1 \circ \ldots \circ r_k \sqsubseteq r$	$r_1^I \circ \ldots \circ r_k^I \subseteq r^I$
transitivity	transitive(r)	$\forall d,e,f \in \Delta^I : (d,e),(e,f) \in r^I \rightarrow (d,f) \in r^I$
reflexivity	reflexive(r)	$\forall d \in \Delta^I : (d,d) \subseteq r^I$
range restriction	range(r)$\sqsubseteq C$	$\{e \in \Delta^I \mid \exists d : (d,e) \in r^I\} \subseteq C^I$
domain restriction	domain(r)$\sqsubseteq C$	$\{d \in \Delta^I \mid \exists e : (d,e) \in r^I\} \subseteq C^I$
role hierarchy (RH)	$r \sqsubseteq s$	$r^I \subseteq s^I$

Table 1: $\mathcal{EL}^+$ syntax and semantics

on description logics, context free grammars, the Earley algorithm, and existing versions of reachability. In Section 3 we introduce a notion of backward (top-down) reachability. We show that a bi-directional reachability-based approach may be used to extract small modules that considers the sub-concept as well as the super-concept in an entailment. We show that modules obtained in this way may be smaller than reachability-based modules that consider only the sub-concept. In Section 4 we provide an algorithm for transforming any $\mathcal{EL}^+$ ontology into a reachability preserving CFG and show how the Earley algorithm can be used to extract a small strong subsumption module for a given entailment. Furthermore, in section 5 we show that the Earley algorithm simultaneously computes all parse trees, where each parse tree corresponds to a possible MinA. Section 6 is a discussion on work in progress, where we discuss possible changes to the standard Earley algorithm specifically aimed at optimizing the MinA discovery process. Section 7 briefly concludes and discusses future work.

2 Preliminaries

2.1 DL terminology

In the standard set-theoretic semantics of concept descriptions, concepts are interpreted as subsets of a domain of interest, and roles as binary relations over this domain. An interpretation I consists of a non-empty set Δ^I (the *domain* of I) and a function $\cdot^I$ (the *interpretation function* of I) which maps each atomic concept A to a subset A^I of Δ^I, and each atomic role R to a subset R^I of $\Delta^I \times \Delta^I$. The interpretation function is extended to arbitrary concept and role descriptions, with the specifics depending on the particular description logic under consideration. We provide the details for $\mathcal{EL}^+$ in Definition 1 below.

A DL knowledge base consists of a *TBox* which contains *terminological axioms*, and an *ABox* which contains *assertions*, i.e. facts about specific named objects and relationships between objects in the domain. For the purposes of this paper we concern ourselves only with Tbox statements.

TBox statements are *general concept inclusions* of the form $C \sqsubseteq D$, where C and D are (possibly complex) concept descriptions. $C \sqsubseteq D$ is also called a *subsumption statement*, read "C is subsumed by D". An interpretation I *satisfies* $C \sqsubseteq D$, written $I \Vdash C \sqsubseteq D$, iff $C^I \subseteq D^I$. $C \sqsubseteq D$ is *valid*, written $\models C \sqsubseteq D$, iff it is satisfied by all interpretations.

An interpretation I satisfies a DL knowledge base $\mathcal{K}$ iff it satisfies every statement in $\mathcal{K}$. A DL knowledge base $\mathcal{K}$ *entails* a DL statement ϕ, written as $\mathcal{K} \models \phi$, iff every interpretation that satisfies $\mathcal{K}$ also satisfies ϕ.

Roughly speaking, DLs are defined by the constructors they provide. There exists a correlation between the expressivity of the DL and the complexity of reasoning over it. We consider the DL $\mathcal{EL}^+$ which is defined as follows:

Definition 1 ($\mathcal{EL}^+$ syntax and semantics) *The syntax and semantics of $\mathcal{EL}^+$ constructors are defined in Table 1.*

We further require that an $\mathcal{EL}^+$ ontology conform to the following ***syntactic restriction*** (Suntisrivaraporn 2009): For an ontology $\mathcal{O}$ and role names r,s, we write $\mathcal{O} \models r \sqsubseteq s$ if and only if $r = s$ or $\mathcal{O}$ contains role inclusions $r_1 \sqsubseteq r_2, \ldots, r_{k-1} \sqsubseteq r_k$ with $r = r_1$ and $s = r_k$. Also, we write $\mathcal{O} \models \mathit{range}(r) \sqsubseteq C$ if there is a role name s such that $\mathcal{O} \models r \sqsubseteq s$ and $\mathit{range}(s) \sqsubseteq C \in \mathcal{O}$. The $\mathcal{EL}^+$ syntactic restriction is as follows: If $r_1 \circ \ldots \circ r_k \sqsubseteq s \in \mathcal{O}$ with $k \geq 1$ and $\mathcal{O} \models \mathit{range}(s) \sqsubseteq C$, then $\mathcal{O} \models \mathit{range}(r_k) \sqsubseteq C$.

Intuitively, the restriction ensures that a role inclusion $r_1 \circ \cdots \circ r_k \sqsubseteq s$ does not induce any new range constraints on the role composition $r_1 \circ \cdots \circ r_k$. Formally, it ensures that if the role inclusion implies a role relationship $(d,\ e) \in s^I$ in the model, then the range restrictions on s do not impose new concept memberships on e. Without this restriction reasoning in $\mathcal{EL}^+$ becomes intractable (Suntisrivaraporn 2009, Baader et al. 2008).

Given an ontology $\mathcal{O}$ and an entailment $\mathcal{O} \models \sigma$ with σ a statement of interest, a *justification* for σ is a set of axioms from $\mathcal{O}$ such that the entailment is preserved. A *minimal axiom set (MinA)* is the smallest set of axioms that preserves the entailment.

Definition 2 (Minimal Axiom set) *Let $\mathcal{O}$ be an ontology, and σ a subsumption statement such that $\mathcal{O} \models \sigma$. A subset $S \subseteq \mathcal{O}$ is a minimal axiom set (MinA) for σ w.r.t. $\mathcal{O}$, also written as "S is a MinA for $\mathcal{O} \models \sigma$", if and only if*

1. *$S \models \sigma$, and*
2. *for every $S' \subset S$, $S' \not\models \sigma$.*

Definition 3 (Signature of $\mathcal{O}$) *Let* CN*($\mathcal{O}$) represent the set of all concept names in $\mathcal{O}$,* RN*($\mathcal{O}$) the set of all role names in $\mathcal{O}$. We define the signature of $\mathcal{O}$, denoted as Sig($\mathcal{O}$), as the union of all concept and role names occurring in $\mathcal{O}$ i.e., Sig($\mathcal{O}$) =* CN*($\mathcal{O}$) ∪* RN*($\mathcal{O}$). Similarly for any $\mathcal{EL}^+$ statement σ, Sig(σ) is the union of all concept and role names occurring in σ.*

2.2 The Earley algorithm for parsing CFG languages

Context free grammars (CFGs) provide a well-known method for modeling the structure of English and other natural languages. A grammar consists of a set of productions or rules, each of which expresses the ways the symbols (strings) in a language can be grouped together, as well as a lexicon of words or symbols.

Definition 4 (CFG production rules) *Given that X represents a single non-terminal, the symbol "a" represents a single terminal and α and σ represent mixed strings of terminals and non-terminals, including the* null *string. CFG production rules have the form:*

$$X \rightarrow \alpha\sigma \quad (1)$$
$$X \rightarrow a \quad (2)$$

Parsing a CFG string results in a parse tree, assigning syntactic structure to it. The Earley parsing algorithm (Earley 1970) uses a dynamic programming approach applying a single left-to-right, top-down, depth-first parallel search strategy to compute a chart that contains all possible parses for a given input. It accomplishes this in polynomial worst case time (n^3), where n is the size of the input string.

Example 1 *Consider the sample CFG for a subset of English grammar below. The set of symbols $\{that, book, flight\}$ represent terminal symbols and all other symbols represent non-terminals.*

$$\begin{aligned} S &\rightarrow VP \\ S &\rightarrow NP\ \ VP \\ VP &\rightarrow VP\ \ NP \\ NP &\rightarrow Det\ \ Noun \\ VP &\rightarrow Verb \\ Det &\rightarrow that \\ Verb &\rightarrow book \\ Noun &\rightarrow flight \end{aligned}$$

During execution the Earley algorithm generates a state entry for each production rule it operates on. The purpose of the state is to record the progress made during the parsing process.

Definition 5 (Parse states) *Let X and Y represent single non-terminal symbols, let a represent a single terminal symbol, and let α, β and σ represent mixed strings of terminal and non-terminal symbols, including the* null *string. Then for each token (word) in the input string, the Earley algorithm creates a set of states, called a chart. A chart at position k of the input is represented by $\mathcal{C}_k$. Each state consists of a tuple $(X \rightarrow \alpha \bullet \beta,\ i)$ where*

1. *$X \rightarrow \alpha\beta$ is the current production rule,*
2. *$\bullet$ indicates the* dot rule *which represents the current parsing position in the state, with $\alpha \bullet \beta$ indicating that α has previously been parsed and β is expected next, and*
3. *i indicates the starting index of the substring where parsing of this production began.*

The parser consists of three sub-parts, the *predictor*, *scanner* and *completer*. For each state in chart $\mathcal{C}_i$, the tuple $(X \rightarrow \alpha \bullet \beta,\ j)$ is evaluated and the appropriate sub-part executed:

1. **Predictor:** If state $= (X \rightarrow \alpha \bullet Y\beta,\ j)$, then for every production $Y \rightarrow \sigma$, if $(Y \rightarrow \bullet\sigma) \notin \mathcal{C}_i$ then $\mathcal{C}_i := \mathcal{C}_i + (Y \rightarrow \bullet\sigma,\ i)$,
2. **Scanner:** If state $= (X \rightarrow \alpha \bullet a\beta,\ j)$, with a the next symbol in the input stream, and if $(X \rightarrow \alpha a \bullet \beta,\ j) \notin \mathcal{C}_{i+1}$ then $\mathcal{C}_{i+1} := \mathcal{C}_{i+1} + (X \rightarrow \alpha a \bullet \beta,\ j)$,
3. **Completer:** If state $= (X \rightarrow \gamma \bullet,\ j)$, then for every $(Y \rightarrow \alpha \bullet X\beta,\ k) \in \mathcal{C}_j$, $\mathcal{C}_i := \mathcal{C}_i + (Y \rightarrow \alpha X \bullet \beta,\ k)$.

The algorithm executes all states iteratively in a top-down manner until no new states are available for processing, and no state may appear more than once in a given chart (Jurafsky & Martin 2009).

Example 2 *Table 2 shows the output of the Earley algorithm given the input string* 'book that flight' *and the CFG in Example 1. The state*

$$20.\quad S \rightarrow VP\bullet$$

in chart 3 represents a successful parse of the string.

Table 2: Parse for 'book that flight'

Chart 0:	• book that flight
1. S → • NP VP	$j = 0$: Inital State
2. S → • VP	$j = 0$: Inital State
3. NP → • Det Noun	$j = 0$: Predictor 1
4. VP → • VP NP	$j = 0$: Predictor 2
5. VP → • Verb	$j = 0$: Predictor 2
6. Det → • that	$j = 0$: Predictor 3
7. Verb → • book	$j = 0$: Predictor 5
Chart 1:	book • that flight
8. Verb → book •	$j = 0$: Scanner 7
9. VP → Verb •	$j = 0$: Completer 5, 8
10. VP → VP • NP	$j = 0$: Completer 4, 8
11. NP → • Det Noun	$j = 1$: Predictor 10
12. Det → • that	$j = 1$: Predictor 11
Chart 2:	book that • flight
13. Det → that •	$j = 1$: Scanner 12
14. NP → Det • Noun	$j = 1$: Completer 11, 13
15. Noun → • flight	$j = 2$: Predictor 14
Chart 3:	book that flight •
16. Noun → flight •	$j = 2$: Scanner 15
17. NP → Det Noun •	$j = 1$: Completer 14, 16
18. VP → VP NP •	$j = 0$: Completer 10, 17
19. VP → VP • NP	$j = 0$: Completer 4, 18
20. S → VP •	$j = 0$: Completer 2, 18

2.3 Reachability-based module extraction

Given an ontology $\mathcal{O}$ and an entailment $\mathcal{O} \models \sigma$ with σ a statement of interest, extracting a module aims to obtain a small subset $\mathcal{O}'$ of $\mathcal{O}$, such that entailment of σ is preserved, where $\mathrm{Sig}(\sigma)$ is defined as in Definition 3. For the purposes of this paper σ is always a subsumption statement.

Definition 6 (Module for $\mathcal{EL}^+$) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, and σ a statement formulated in $\mathcal{EL}^+$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a* module for σ in $\mathcal{O}$*(a σ-*module in $\mathcal{O}$*) whenever: $\mathcal{O} \models \sigma$ if and only if $\mathcal{O}' \models \sigma$. We say that $\mathcal{O}'$ is a module for a signature $\boldsymbol{S}$ in $\mathcal{O}$ (an* **S**-module in $\mathcal{O}$*) if, for every $\mathcal{EL}^+$ statement σ with $Sig(\sigma) \subseteq \boldsymbol{S}$, $\mathcal{O}'$ is a σ-module in $\mathcal{O}$.*

Definition 7 (Reachability-based modules) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. The set of S-reachable names in $\mathcal{O}$ is defined inductively as:*

2.2 The Earley algorithm for parsing CFG languages

Context free grammars (CFGs) provide a well-known method for modeling the structure of English and other natural languages. A grammar consists of a set of productions or rules, each of which expresses the ways the symbols (strings) in a language can be grouped together, as well as a lexicon of words or symbols.

Definition 4 (CFG production rules) *Given that X represents a single non-terminal, the symbol "a" represents a single terminal and α and σ represent mixed strings of terminals and non-terminals, including the* null *string. CFG production rules have the form:*

$$X \rightarrow \alpha\sigma \quad (1)$$
$$X \rightarrow a \quad (2)$$

Parsing a CFG string results in a parse tree, assigning syntactic structure to it. The Earley parsing algorithm (Earley 1970) uses a dynamic programming approach applying a single left-to-right, top-down, depth-first parallel search strategy to compute a chart that contains all possible parses for a given input. It accomplishes this in polynomial worst case time (n^3), where n is the size of the input string.

Example 1 *Consider the sample CFG for a subset of English grammar below. The set of symbols $\{that, book, flight\}$ represent terminal symbols and all other symbols represent non-terminals.*

$$\begin{aligned} S &\rightarrow VP \\ S &\rightarrow NP\ \ VP \\ VP &\rightarrow VP\ \ NP \\ NP &\rightarrow Det\ \ Noun \\ VP &\rightarrow Verb \\ Det &\rightarrow that \\ Verb &\rightarrow book \\ Noun &\rightarrow flight \end{aligned}$$

During execution the Earley algorithm generates a state entry for each production rule it operates on. The purpose of the state is to record the progress made during the parsing process.

Definition 5 (Parse states) *Let X and Y represent single non-terminal symbols, let a represent a single terminal symbol, and let α, β and σ represent mixed strings of terminal and non-terminal symbols, including the* null *string. Then for each token (word) in the input string, the Earley algorithm creates a set of states, called a chart. A chart at position k of the input is represented by $\mathcal{C}_k$. Each state consists of a tuple $(X \rightarrow \alpha \bullet \beta,\ i)$ where*

1. *$X \rightarrow \alpha\beta$ is the current production rule,*
2. *$\bullet$ indicates the* dot rule *which represents the current parsing position in the state, with $\alpha \bullet \beta$ indicating that α has previously been parsed and β is expected next, and*
3. *i indicates the starting index of the substring where parsing of this production began.*

The parser consists of three sub-parts, the *predictor*, *scanner* and *completer*. For each state in chart $\mathcal{C}_i$, the tuple $(X \rightarrow \alpha \bullet \beta,\ j)$ is evaluated and the appropriate sub-part executed:

1. **Predictor:** If state $= (X \rightarrow \alpha \bullet Y\beta,\ j)$, then for every production $Y \rightarrow \sigma$, if $(Y \rightarrow \bullet\sigma) \notin \mathcal{C}_i$ then $\mathcal{C}_i := \mathcal{C}_i + (Y \rightarrow \bullet\sigma,\ i)$,
2. **Scanner:** If state $= (X \rightarrow \alpha \bullet a\beta,\ j)$, with a the next symbol in the input stream, and if $(X \rightarrow \alpha a \bullet \beta,\ j) \notin \mathcal{C}_{i+1}$ then $\mathcal{C}_{i+1} := \mathcal{C}_{i+1} + (X \rightarrow \alpha a \bullet \beta,\ j)$,
3. **Completer:** If state $= (X \rightarrow \gamma\bullet,\ j)$, then for every $(Y \rightarrow \alpha \bullet X\beta,\ k) \in \mathcal{C}_j$, $\mathcal{C}_i := \mathcal{C}_i + (Y \rightarrow \alpha X \bullet \beta,\ k)$.

The algorithm executes all states iteratively in a top-down manner until no new states are available for processing, and no state may appear more than once in a given chart (Jurafsky & Martin 2009).

Example 2 *Table 2 shows the output of the Earley algorithm given the input string* 'book that flight' *and the CFG in Example 1. The state*

$$20.\quad S \rightarrow VP\bullet$$

in chart 3 represents a successful parse of the string.

Table 2: Parse for 'book that flight'

Chart 0:	• book that flight
1. S → • NP VP	$j = 0$: Inital State
2. S → • VP	$j = 0$: Inital State
3. NP → • Det Noun	$j = 0$: Predictor 1
4. VP → • VP NP	$j = 0$: Predictor 2
5. VP → • Verb	$j = 0$: Predictor 2
6. Det → • that	$j = 0$: Predictor 3
7. Verb → • book	$j = 0$: Predictor 5
Chart 1:	book • that flight
8. Verb → book •	$j = 0$: Scanner 7
9. VP → Verb •	$j = 0$: Completer 5, 8
10. VP → VP • NP	$j = 0$: Completer 4, 8
11. NP → • Det Noun	$j = 1$: Predictor 10
12. Det → • that	$j = 1$: Predictor 11
Chart 2:	book that • flight
13. Det → that •	$j = 1$: Scanner 12
14. NP → Det • Noun	$j = 1$: Completer 11, 13
15. Noun → • flight	$j = 2$: Predictor 14
Chart 3:	book that flight •
16. Noun → flight •	$j = 2$: Scanner 15
17. NP → Det Noun •	$j = 1$: Completer 14, 16
18. VP → VP NP •	$j = 0$: Completer 10, 17
19. VP → VP • NP	$j = 0$: Completer 4, 18
20. S → VP •	$j = 0$: Completer 2, 18

2.3 Reachability-based module extraction

Given an ontology $\mathcal{O}$ and an entailment $\mathcal{O} \models \sigma$ with σ a statement of interest, extracting a module aims to obtain a small subset $\mathcal{O}'$ of $\mathcal{O}$, such that entailment of σ is preserved, where $\mathrm{Sig}(\sigma)$ is defined as in Definition 3. For the purposes of this paper σ is always a subsumption statement.

Definition 6 (Module for $\mathcal{EL}^+$) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, and σ a statement formulated in $\mathcal{EL}^+$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a* module for σ in $\mathcal{O}$*(a σ-*module in $\mathcal{O}$*) whenever: $\mathcal{O} \models \sigma$ if and only if $\mathcal{O}' \models \sigma$. We say that $\mathcal{O}'$ is a module for a signature $\boldsymbol{S}$ in $\mathcal{O}$ (an* $\boldsymbol{S}$-module in $\mathcal{O}$*) if, for every $\mathcal{EL}^+$ statement σ with $Sig(\sigma) \subseteq \boldsymbol{S}$, $\mathcal{O}'$ is a σ-module in $\mathcal{O}$.*

Definition 7 (Reachability-based modules) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. The set of S-reachable names in $\mathcal{O}$ is defined inductively as:*

- *x is S-reachable in $\mathcal{O}$, for every $x \in S$;*
- *for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R$, if x is S-reachable in $\mathcal{O}$ for every $x \in Sig(\alpha_L)$, then y is S-reachable in $\mathcal{O}$ for every $y \in Sig(\alpha_R)$.*

We call an axiom $\alpha_L \sqsubseteq \alpha_R$ S-reachable in $\mathcal{O}$ if every element of $Sig(\alpha_L)$ is S-reachable in $\mathcal{O}$. The reachability-based module for S in $\mathcal{O}$, denoted by $\mathcal{O}_S^{reach}$, consists of all S-reachable axioms from $\mathcal{O}$.

When S is the single concept A, we write A-reachable and $\mathcal{O}_A^{reach}$. An interesting result of reachability is that it can be used to test negative subsumption. That is, if B is not A-reachable in $\mathcal{O}$, then $\mathcal{O} \not\models A \sqsubseteq B$, unless A is unsatisfiable w.r.t $\mathcal{O}$ (Suntisrivaraporn 2009).

Definition 8 (Subsumption module) *Let $\mathcal{O}$ be an ontology, and A a concept name occurring in $\mathcal{O}$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a subsumption module for A in $\mathcal{O}$ whenever: $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}' \models A \sqsubseteq B$ holds for every concept name B occurring in $\mathcal{O}$.*

A subsumption module $\mathcal{O}'$ for A in $\mathcal{O}$ is called strong *if the following holds for every concept name B occurring in $\mathcal{O}$: if $\mathcal{O} \models A \sqsubseteq B$, then every MinA for $\mathcal{O} \models A \sqsubseteq B$ is a subset of $\mathcal{O}'$.*

Theorem 1 *(Suntisrivaraporn 2009). The module $\mathcal{O}_A^{reach}$ is a strong subsumption module for A in $\mathcal{O}$.*

We require that an $\mathcal{EL}^+$ ontology $\mathcal{O}$ be in normal form. We use the same form as Brandt (2004) and Suntisrivaraporn (2009). Any $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be converted to an ontology $\mathcal{O}'$ in normal form in linear time, with at most a linear increase in the size of the ontology.

Let $CN(\mathcal{O})$ represent the set of all concept names in $\mathcal{O}$, $RN(\mathcal{O})$ the set of all role names in $\mathcal{O}$, $CN(\mathcal{O})^\top = CN(\mathcal{O}) \cup \{\top\}$ and $CN(\mathcal{O})^\bot = CN(\mathcal{O}) \cup \{\bot\}$.

Definition 9 (Normal Form) *An $\mathcal{EL}^+$ ontology $\mathcal{O}$ is in* normal form *if the following conditions are satisfied:*

1. *all concept inclusions in $\mathcal{O}$ have one of the following forms:*
$$\begin{aligned} A_1 \sqcap \ldots \sqcap A_n &\sqsubseteq B, \\ A_1 &\sqsubseteq \exists r.A_2, \\ \exists r.A_1 &\sqsubseteq B \end{aligned}$$
where $A_i \in CN^\top(\mathcal{O})$ and $B \in CN^\bot(\mathcal{O})$;
2. *all role inclusions in $\mathcal{O}$ have one of the following forms:*
$$\begin{aligned} \epsilon &\sqsubseteq r, \\ r &\sqsubseteq s, \\ r \circ s &\sqsubseteq t, \end{aligned}$$
where $r, s, t \in RN(\mathcal{O})$ and ϵ is the identity element;
3. *there are no reflexivity statements, transitivity statements or domain restrictions, and all range restrictions are of the form range$(r) \sqsubseteq A$ with A a concept name.*

3 Bi-directional reachability-based module

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$ and entailment $\mathcal{O} \models A \sqsubseteq B$, as well as the module $\mathcal{O}_A^{reach}$, we have that $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}_A^{reach} \models A \sqsubseteq B$, where A and B are concepts names. $\mathcal{O}_A^{reach}$ preserves entailments for all concept names α such that $\mathcal{O} \models A \sqsubseteq \alpha$.

A criticism raised against reachability-based modules, in terms of finding justifications, is that they contain many irrelevant axioms, and in some cases do not reduce the size of the ontology at all (Jianfeng Du & Ji 2009). This stems from the fact that $\mathcal{O}_A^{reach}$ considers only the sub-concept A in $\mathcal{O} \models A \sqsubseteq B$; the super-concept B is never used to elimate unwanted axioms.

Example 3 *Given the small ontology $\mathcal{O}$ below, as well as $\mathcal{O} \models A \sqsubseteq B$, $\mathcal{O}_A^{reach}$ will consist of axioms 1, 2 and 4. Axiom 4 is irrelevant in terms of finding justifications for $\mathcal{O} \models A \sqsubseteq B$, yet it is included in $\mathcal{O}_A^{reach}$.*

$$\begin{aligned} A &\sqsubseteq \exists r.D && (1) \\ \exists r.D &\sqsubseteq B && (2) \\ E &\sqsubseteq B && (3) \\ A &\sqsubseteq F && (4) \end{aligned}$$

Given the entailment $\mathcal{O} \models A \sqsubseteq B$, reachability can be applied in two directions: The standard bottom-up approach, which extracts $\mathcal{O}_A^{reach}$, and a top-down approach, which extracts $\mathcal{O}_{\overleftarrow{B}}^{reach}$, and is defined as follows:

Definition 10 (Top-down reachability-based module) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. The set of $\overleftarrow{S}$-reachable names in $\mathcal{O}$ is defined inductively as:*

- *x is $\overleftarrow{S}$-reachable in $\mathcal{O}$, for every $x \in S$;*
- *for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R$, if x is $\overleftarrow{S}$-reachable in $\mathcal{O}$ for some $x \in Sig(\alpha_R)$, or if $\alpha_R = \bot$, then y is $\overleftarrow{S}$-reachable in $\mathcal{O}$ for every $y \in Sig(\alpha_L)$.*

We call an axiom $\alpha_L \sqsubseteq \alpha_R$ $\overleftarrow{S}$-reachable in $\mathcal{O}$ if some element of $Sig(\alpha_R)$ is $\overleftarrow{S}$-reachable or if $\alpha_R = \bot$. The top-down reachability-based module for S in $\mathcal{O}$, denoted by $\mathcal{O}_{\overleftarrow{S}}^{reach}$, consists of all $\overleftarrow{S}$-reachable axioms from $\mathcal{O}$.

Besides the direction of application, there is a fundamental difference between the two approaches. When extracting $\mathcal{O}_A^{reach}$, the axiom $\alpha_L \sqsubseteq \alpha_R$ becomes A-reachable only when all $x_i \in \text{Sig}(\alpha_L)$ are A-reachable. When extracting $\mathcal{O}_{\overleftarrow{B}}^{reach}$, the axiom $\alpha_L \sqsubseteq \alpha_R$ is $\overleftarrow{B}$-reachable whenever any $x_i \in \text{Sig}(\alpha_R)$ is $\overleftarrow{B}$-reachable.

By definition of reachability, axioms of the form $\top \sqsubseteq \alpha_R$ and $\epsilon \sqsubseteq r$ are A-reachable, since $\text{Sig}(\top) = \text{Sig}(\epsilon) = \emptyset$, and form part of any module $\mathcal{O}_A^{reach}$ extracted (Suntisrivaraporn 2009).

Further by definition of top-down reachability, axioms of the form $\alpha_L \sqsubseteq \bot$ are $\overleftarrow{B}$-reachable. Therefore all axioms $\alpha_L \sqsubseteq \bot$ will also always be a part of any module $\mathcal{O}_{\overleftarrow{B}}^{reach}$ being extracted.

From Theorem 1 we have that $\mathcal{O}_A^{reach}$ preserves all entailments in terms of the sub-concept A. We show in Theorem 2 that a similar result holds for $\mathcal{O}_{\overleftarrow{B}}^{reach}$ with respect to the super-concept B.

Definition 11 (Top-down subsumption module) *Let $\mathcal{O}$ be an ontology, and B a concept name occurring in $\mathcal{O}$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a top-down subsumption module for B in $\mathcal{O}$ whenever: $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}' \models A \sqsubseteq B$ holds for every concept name A occurring in $\mathcal{O}$.*

A top-down subsumption module $\mathcal{O}'$ for B in $\mathcal{O}$ is called strong *if the following holds for every concept name A occurring in $\mathcal{O}$: if $\mathcal{O} \models A \sqsubseteq B$, then every MinA for $\mathcal{O} \models A \sqsubseteq B$ is a subset of $\mathcal{O}'$.*

We show that top-down reachability modules preserves all subsumption relationships i.t.o super-concepts.

Lemma 1 *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. Then, $\mathcal{O} \models C \sqsubseteq D$ if and only if $\mathcal{O}^{reach}_{\overline{S}} \models C \sqsubseteq D$ for arbitrary $\mathcal{EL}^+$ concept descriptions C and D such that $Sig(D) \subseteq S$.*

Proof: We have to prove two parts. First: If $\mathcal{O}^{reach}_{\overline{S}} \models C \sqsubseteq D$ then $\mathcal{O} \models C \sqsubseteq D$. This follows directly from the fact that $\mathcal{O}^{reach}_{\overline{S}} \subseteq \mathcal{O}$ and that $\mathcal{EL}^+$ is monotonic.

Second, we show that, if $\mathcal{O} \models C \sqsubseteq D$ then $\mathcal{O}^{reach}_{\overline{S}} \models C \sqsubseteq D$: Assume the contrary, that is, assume $\mathcal{O} \models C \sqsubseteq D$ but that $\mathcal{O}^{reach}_{\overline{S}} \not\models C \sqsubseteq D$. Then there must exist an interpretation I and an individual $w \in \Delta^I$ such that I is a model of $\mathcal{O}^{reach}_{\overline{S}}$ and $w \in C^I \setminus D^I$. Modify I to I' by setting $x^{I'} := \Delta^I$ for all concept names $x \in \text{Sig}(\mathcal{O}) \setminus (S \cup \text{Sig}(\mathcal{O}^{reach}_{\overline{S}}))$, and $r^{I'} := \Delta^I \times \Delta^I$ for all roles names $r \in \text{Sig}(\mathcal{O}) \setminus (S \cup \text{Sig}(\mathcal{O}^{reach}_{\overline{S}}))$. I' is a model of $\mathcal{O}^{reach}_{\overline{S}}$ since it does not change the interpretation of any symbol in its signature. For each $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O} \setminus \mathcal{O}^{reach}_{\overline{S}}$, we have $\alpha_L^{I'} \subseteq \alpha_R^{I'}$ since α is not $\overleftarrow{S}$-reachable and thus $\alpha_R^{I'} = \Delta^I$. Therefore I' is a model for $\mathcal{O}$. But I and I' correspond on all symbols $y \in \text{Sig}(D) \subseteq S$ and $C^I \subseteq C^{I'}$, therefore we have that $w \in C^{I'} \setminus D^{I'}$, contradicting the assumption.

In order to show that $\mathcal{O}^{reach}_{\overline{B}}$ contains all MinAs for the entailment $\mathcal{O} \models A \sqsubseteq B$, we show that $\mathcal{O}^{reach}_{\overline{B}}$ is a strong top-down subsumption module:

Theorem 2 *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and B a concept name occurring in $\mathcal{O}$. Then $\mathcal{O}^{reach}_{\overline{B}}$ is a strong top-down subsumption module for B in $\mathcal{O}$.*

Proof: That $\mathcal{O}^{reach}_{\overline{B}}$ is a top-down subsumption module follows directly from Lemma 1 above. To show that it is strong, assume that $\mathcal{O} \models A \sqsubseteq B$, but there is a MinA S for $\mathcal{O} \models A \sqsubseteq B$ that is not contained in $\mathcal{O}^{reach}_{\overline{B}}$. Thus, there must be an axiom $\alpha \in S \setminus \mathcal{O}^{reach}_{\overline{B}}$. Define $S_1 := S \cap \mathcal{O}^{reach}_{\overline{B}}$. S_1 is a strict subset of S since $\alpha \notin S_1$. We claim that $S_1 \models A \sqsubseteq B$, which contradicts the fact that S is a MinA for $\mathcal{O} \models A \sqsubseteq B$.

We use proof by contradiction to show this. Assume that $S_1 \not\models A \sqsubseteq B$ i.e., there is a model I_1 of S_1 such that $A^{I_1} \not\subseteq B^{I_1}$. We modify I_1 to I by setting $y^I := \Delta^{I_1}$ for all concept names y that are not $\overleftarrow{B}$-reachable, and $r^I := \Delta^{I_1} \times \Delta^{I_1}$ for all roles names r that are not $\overleftarrow{B}$-reachable. We have $B^I = B^{I_1}$ since B is $\overleftarrow{B}$-reachable, and $A^I = A^{I_1}$ if A is $\overleftarrow{B}$-reachable, or $A^I = \Delta^{I_1}$ otherwise. Therefore $A^I \not\subseteq B^I$. It remains to be shown that I is indeed a model of S, and therefore satisfies all axioms $\beta = (\beta_L \sqsubseteq \beta_R)$ in S, including $A \sqsubseteq B$. There are two possibilities:

- $\beta \in S_1$. Since $S_1 \subseteq \mathcal{O}^{reach}_{\overline{B}}$, all symbols in $\text{Sig}(\beta_L)$ and one or more symbols in $\text{Sig}(\beta_R)$ are $\overleftarrow{B}$-reachable. Consequently, I_1 and I coincide on the names occurring in β_L and since I_1 is a model of S_1, we have that $(\beta_L)^I = (\beta_L)^{I_1}$ and $(\beta_R)^{I_1} \subseteq (\beta_R)^I$. Therefore $(\beta_L)^I \subseteq (\beta_R)^I$.
- $\beta \notin S_1$. Since $S_1 = S \setminus \mathcal{O}^{reach}_{\overline{B}}$, we have that β is not $\overleftarrow{B}$-reachable. Thus no $x \in \text{Sig}(\beta_R)$ is $\overleftarrow{B}$-reachable. By the definition of I, $(\beta_R)^I = \Delta^{I_1}$. Hence $(\beta_L)^I \subseteq (\beta_R)^I$.

Therefore I is a model for S.

Example 4 *Extracting $\mathcal{O}^{reach}_{\overline{B}}$ from the sample ontology in Example 3, we see that it will consist of axioms 1, 2 and 3. This correctly differs from $\mathcal{O}^{reach}_{A}$ in that axiom 4 is not $\overleftarrow{B}$-reachable. Similar to $\mathcal{O}^{reach}_{A}$ though, it contains an axiom that is irrelevant in terms of finding justifications for $\mathcal{O} \models A \sqsubseteq B$, axiom 3 in this case.*

It is clear that when extracting modules for finding justifications, $\mathcal{O}^{reach}_{\overline{B}}$ opens itself to the same criticism as $\mathcal{O}^{reach}_{A}$. Given the entailment $\mathcal{O} \models A \sqsubseteq B$ we see that where $\mathcal{O}^{reach}_{A}$ considers only the sub-concept A, $\mathcal{O}^{reach}_{\overline{B}}$ considers only the super-concept B.

Both reachability module extraction methods preserve all entailments; $\mathcal{O}^{reach}_{A}$ entailments in terms of the sub-concept A and $\mathcal{O}^{reach}_{\overline{B}}$ entailments in terms of the super-concept B. Given the entailment $\mathcal{O} \models A \sqsubseteq B$, we may now extract the module $(\mathcal{O}^{reach}_{A})^{reach}_{\overline{B}}$, or similarly $(\mathcal{O}^{reach}_{\overline{B}})^{reach}_{A}$, such that it considers both the sub- and super concepts in the entailment. The resulting module is a *bi-directional reachability-based module* denoted by $\mathcal{O}^{reach}_{A \leftrightarrow B}$.

Definition 12 (Bi-directional reachability-based module) *A bi-directional reachability-based module, denoted $\mathcal{O}^{reach}_{A \leftrightarrow B}$, is defined as the set of all axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ such that for every $x_i \in Sig(\alpha_L)$, x_i is A-reachable, and for some $y_i \in Sig(\alpha_R)$, y_i is $\overleftarrow{B}$-reachable.*

Example 5 *Extracting $\mathcal{O}^{reach}_{A \leftrightarrow B}$ from the sample ontology in Example 3, we see that it will consist of axioms 1 and 2. The previous irrelevant axioms 3 and 4 are no longer present.*

From Theorem 1 and Theorem 2 we have that bi-derectional reachabiliy modules preserves all MinAs for the entailment $\mathcal{O} \models A \sqsubseteq B$.

Corollary 1 *$\mathcal{O}^{reach}_{A \leftrightarrow B}$ preserves all MinAs for $\mathcal{O} \models A \sqsubseteq B$.*

4 Reachability preserving CFG

We show how an $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be transformed into a reachability preserving CFG. In the discussion that follows we assume that we have an $\mathcal{EL}^+$ ontology $\mathcal{O}$ in normal form, and an entailment $\mathcal{O} \models A \sqsubseteq B$, where A and B are single concept names.

From the definition of $\mathcal{O}^{reach}_{A \leftrightarrow B}$ above , we have that every axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}^{reach}_{A \leftrightarrow B}$ has the following two properties:

1. every $x_i \in \text{Sig}(\alpha_L)$ is A-reachable, and
2. some $y_i \in \text{Sig}(\alpha_R)$ is $\overleftarrow{B}$-reachable.

Every CFG production rule we introduce must preserve bi-directional reachability. By Property 1 above, A-reachability of the axiom $\alpha_L \sqsubseteq \alpha_R$ is solely dependent on symbols in α_L. Similarly, by Property 2, $\overleftarrow{B}$-reachability is solely dependent on symbols in α_R.

From the previous section we know that there exists special cases in which A- and $\overleftarrow{B}$-reachability hold. For A-reachability these axioms have one of the forms:

- $\top \sqsubseteq \alpha_R$, or
- $c \sqsubseteq \alpha_R$

For $\overleftarrow{\text{B}}$-reachability these axioms have the form:

- $\alpha_L \sqsubseteq \bot$

In the steps that follow, all production rules we introduce have the form $y_i \rightarrow \sigma$. Each rule is read as: any $\overleftarrow{\text{B}}$-reachable symbol y_i is A-reachable only if all symbols $x_i \in \sigma$ are A-reachable. This clearly conforms to the definition of bi-directional reachability. We further note that the symbols on the rhs of CFG production rules have a fixed order, whereas the conjunction of $\mathcal{EL}^+$ concepts and roles are symmetric, i.e. $A \sqcap B = B \sqcap A$, and thus order is unimportant. We therefore place no restrictions on the order of the symbols on the rhs of production rules and thus consider production rules differing only in the order of symbols on the rhs as identical.

The conversion process below proceeds in a step by step manner until all axioms in $\mathcal{O}$ have been processed.

Step 1: All axioms $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ such that $\text{Sig}(\alpha_R) = \emptyset$ are $\overleftarrow{\text{B}}$-reachable by definition. By Property 2 above, in order to preserve both $\overleftarrow{\text{B}}$-reachability as well as bi-directional reachability, $\overleftarrow{\text{B}}$-reachability depends solely on α_R. For each such axiom the implicit $\overleftarrow{\text{B}}$-reachability of α_R is made explicit by introducing the following production rule:

$$B \rightarrow Sig(\alpha_L)$$

Step 2: All axioms $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ such that $\text{Sig}(\alpha_L) = \emptyset$ are (implicitly) A-reachable. By property 1 above, in order to preserve both A-reachability as well as bi-directional reachability, A-reachability depends solely on α_L. For each such axiom the implicit A-reachability of α_L is made explicit by introducing the production rule:

$$y_i \rightarrow A$$

for each $y_i \in Sig(\alpha_R)$.

Step 3: For each axiom $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ such that $|\text{Sig}(\alpha_L)| \geq 1$ and $|\text{Sig}(\alpha_R)| \geq 1$, introduce the production rule:

$$y_i \rightarrow Sig(\alpha_L)$$

for each $y_i \in Sig(\alpha_R)$. Axioms of this kind do not have any implicit reachability concerns like those in Steps 1 and 2 above, and bi-directional reachability is preserved trivially.

We note that it follows from the normal form in Definition 9 that, for every rule introduced in Steps 2 and 3 above where $|\text{Sig}(\alpha_R)| = 2$, α_R has the form $\exists r.C$. By property 2 above, $\overleftarrow{\text{B}}$-reachability is preserved if either one of r or C is $\overleftarrow{\text{B}}$-reachable. Bi-directional reachability is therefore preserved by the two rules:

$$r \rightarrow Sig(\alpha_L)$$

$$C \rightarrow Sig(\alpha_L)$$

in Step 3 above, and similarly for Step 2. We therefore define the reachability preserving CFG for an $\mathcal{EL}^+$ ontology $\mathcal{O}$ as:

Definition 13 (Reachability preserving CFG) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form, and $\mathcal{O} \models A \sqsubseteq B$ an entailment, then the reachability preserving CFG, denoted $\mathcal{CFG}_\mathcal{O}$, is the minimal set of CFG production rules such that for each axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$:*

- *if $Sig(\alpha_R) = \emptyset$, the rule $B \rightarrow Sig(\alpha_L) \in \mathcal{CFG}_\mathcal{O}$;*
- *if $Sig(\alpha_L) = \emptyset$ the rule $x_i \rightarrow A \in \mathcal{CFG}_\mathcal{O}$ for each $x_i \in Sig(\alpha_R)$;*
- *for all other axioms the rule $x_i \rightarrow Sig(\alpha_L) \in \mathcal{CFG}_\mathcal{O}$ for each $x_i \in Sig(\alpha_R)$;*

where the symbol A represents the only terminal symbol and the set $Sig(\mathcal{O})\backslash A$ represent the set of non-terminals.

Example 6 *All production rules for $\mathcal{CFG}_\mathcal{O}$ may be obtained from the ontology in Example 3 above as follows:*

r	$\rightarrow$	A	*(Step 3 applied to axiom 1)*
D	$\rightarrow$	A	*(Step 3 applied to axiom 1)*
B	$\rightarrow$	$r\ D$	*(Step 3 applied to axiom 2)*
B	$\rightarrow$	E	*(Step 3 applied to axiom 3)*
F	$\rightarrow$	A	*(Step 3 applied to axiom 4)*

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$, with $\mathcal{O}'$ being $\mathcal{O}$ in normal form, and n the number of axioms in $\mathcal{O}'$, we see that there can be at most $2 \times n$ production rules in $CFG_{\mathcal{O}'}$. This follows directly from the definition of $CFG_{\mathcal{O}'}$, and the fact that there is a one-to-one correspondence between all CFG production rules introduced and axioms in $\mathcal{O}'$, except for axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}'$ where $\alpha_R = \exists r.C$; for these axioms two production rules are introduced.

5 Earley as a MinA extraction algorithm

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$, an entailment $\mathcal{O} \models A \sqsubseteq B$ and the resulting CFG $\mathcal{CFG}_\mathcal{O}$, the Earley algorithm may be applied to extract all possible parse trees.

Before the algorithm is executed we introduce the start state $S \rightarrow B$, where S $\notin$ Sig($\mathcal{O}$). The algorithm now proceeds in a top-down manner, starting with this state, and then proceeds to find all production rules $\sigma_L \rightarrow \sigma_R$ such that σ_L is $\overleftarrow{\text{B}}$-reachable and all $x_i \in \sigma_R$ are A-reachable.

From the definition of $\mathcal{CFG}_\mathcal{O}$ we have that the symbol A is the only terminal symbol and all other symbols are considered to be non-terminals. The standard algorithm requires an input string to parse. While in terms of reachability there is no explicit input string, it is implicit from the definition of $\mathcal{CFG}_\mathcal{O}$ that the input string consists of a finite string of As.

The standard Earley algorithm may now be applied in order to extract all possible reachability paths between the concepts A and B. However, since the Earley algorithm is not explicitly bound by an input string, the situation arises that it may never terminate. The algorithm inherently handles cycles, but the absence of an explicit input-sentence may lead to non-termination. This occurs when a cycle is right-recursive, where the recursive part is not A-reachable. For example let the rules:

$$B \rightarrow C\ D$$

$$C \rightarrow A$$

$$D \rightarrow B$$

represent such a cycle. Then D can never be A-reachable, however since C is A-reachable the Earley algorithm will go into an infinite loop.

There are a few ways to remedy this. The first is to introduce a depth bound n, where n is the sum total of production rulesl symbols appearing on the rhs of production rules. The choice of n stems from the fact that the Earley algorithm exhaustively searches

for parse trees, each node represented by some production rule. In chart k, the Early algorithm searches trees with the length of the longest branch in the tree equal to $k+1$. This branch then represents the longest chain of bi-directional reachability preserving production rules, with each rule appearing at most j times, where j is the sum total of times the lhs symbols of the production rule appears on the rhs of other production rules. Thus the longest branch of any parse tree cannot exceed n. The standard Earley algorithm will in this case run in $\mathcal{O}(n^3)$ worst case time.

A different approach is to first extract $\mathcal{O}_A^{reach}$ and running the Earley algorithm on it. This guarantees that all production rules are A-reachable and the above cycle can never occur, this however does not gurantee that no other bad cycles exists. A third approach involves changing the algorithm itself similar to the method used in Section 6.

Both $\mathcal{O}_A^{reach}$ and $\mathcal{O}_{\overline{B}}^{reach}$ can be extracted in linear time, and hence so can $\mathcal{O}_{A \leftrightarrow B}^{reach}$. The standard Earley algorithm is therefore non-optimal by two orders of magnitude in terms of module extraction. The benefit gained from the Earley algorithm however is that all parse trees are computed simultaneously.

Each parse tree computed by the Earley algorithm corresponds to a set of production rules, starting with the state $S \rightarrow B$, such that for each rule $\sigma_L \rightarrow \sigma_R$, we have that σ_L is $^{\leftarrow}\overline{B}$-reachable and σ_R is A-reachable. Each branch of a parse tree corresponds to a minimal set of productions rules such that B is A-reachable and A is $^{\leftarrow}\overline{B}$-reachable, removing any rule from this set would cause reachability to be lost for that branch, and hence the whole tree would not preserve bi-directional reahcability. Each parse tree therefore corresponds to a possible MinA, dependent only upon a positive subsumption test.

It must be noted that in the worst case, there is an exponential number of parse trees. The Earley algorithm computes all parse trees in parallel in polynomial time. However, extracting an exponential number of parse trees will run in exponential time.

6 Work in progress

In this section we outline some modifications to the Earley algortihm to improve its efficiency in terms of MinA extraction.

During its search for parse trees, the predictor procedure expands all production rules for a non-terminal symbol it encounters. Terminal symbols are not expanded and are handled by the scanner procedure. We note that, in our case, when a concept C becomes A-reachable, future expansions of production rules for C are unneccesary. When a specific reachability path between the concept A and C has been found, we never need to traverse that path again, and the symbol C effectively becomes a terminal symbol.

The algorithm may therefore be improved by introducing a dynamic terminal set. That is, initially only the symbol A is a terminal symbol. When any symbol becomes A-reachable we add it to the set of terminals and remove it from the set of non-terminals.

The completer procedure forms the core of marking parse trees. It keeps track of the production rules responsible for completions; for every symbol in a production rule, it maintains a list of pointers to other states responsible for completing it. Having a dynamic terminal set complicates this bookkeeping process and requires changes to the data-structures used, as well as the completer and predictor procedures.

1. **Data structures:** We introduce an array such that, for each concept/symbol that becomes A-reachable, we maintain a set of pointers to states, responsible for completing the symbol. Each pointer records the state in which the symbol becomes A-reachable, i.e. whenever the completer is run, a pointer to this state entry is recorded in the array of pointers for the symbol being completed.

2. **Predictor:** The predictor procedure normally expands all relevant production rules for a symbol and adds new states to the current chart. It never adds the same state more than once to the current chart. In a different chart however it may expand the same symbol again. We restrict the procedure so that it may never introduce a production rule more than once, irrespective of the chart it which it occurs.

3. **Completer:** For each state completed the completer stores a pointer to the state in the array above for the symbol being completed. Every symbol completed also gets marked as a terminal symbol. The changes in the predictor further neccesitates that once a new symbol D becomes a terminal, that the completer completes all states $\alpha \rightarrow \sigma \bullet D$ in any prior chart, and that the scanner be called for each state $\alpha \rightarrow \sigma \bullet D$ in the current chart.

4. **Production states:** Production states no longer require an index to mark their originating charts, because, for each state, the new completer procedure will scan all previous charts for symbols to the right of the dot to complete, and not only those from the chart the state originated from.

These optimizations have the potential for a more efficient algorithm. The problem of non-termination described earlier is no longer relevant, since every production rule can only ever be introduced once by the predictor. From this we have that the only way a production rule occurs more than once in any chart is by virtue of the scanner or completer procedures. Each of these advances the dot in some way and new symbols may need to be expanded, but these expansions can only be done by the predictor which would never expand any production rule more than once.

The proposed modified Early algorithm is listed in Table 3. Before the algorithm is executed we obtain $\mathcal{CFG}_{\mathcal{O}}$. For the entailment $\mathcal{O} \models A \sqsubseteq B$ the appropriate substitutions have been made and the start state $S \rightarrow B$ added to $\mathcal{CFG}_{\mathcal{O}}$ and chart[0]. The set TERMINALS represent the set of all terminals, initialised to $\{A\}$, NONTERMINALS the set of non-terminals initialised to $\text{Sig}(\mathcal{O}) \backslash A$, and REF[$\alpha$] an initially empty array which will contain pointers to all states where the symbol α has been completed.

Once the algorithm terminates all MinAs still need to be extracted. The process is similar to the method used to extract the parse trees from the original Earley algorithm. The algorithm proceeds in a standard depth-first manner. Starting with the completion references for the symbol S, select the production rule referenced. Let this state be $S \rightarrow \alpha$. Then for each symbol $x_i \in \alpha$ choose a production rule from $\text{REF}(x_i)$; this process continues recursively. Once no new production rules can be added, the set of all states represent a possible MinA. Mapping back to the origional axioms in normal form the set can be tested for subsumption, and if subsumption holds the MinA is valid. More parse trees may be extracted by backtracking and making alternate choices where $|\text{REF}(x_i)| > 1$.

We use the standard example in the literature (Brandt 2004), showing that there exists an ontology $\mathcal{O}$ such that it contains exponentially many MinAs

Table 3: Modified Earley algorithm

```
function EARLY-PARSE returns chart
cIndex = 0
do
  for each state in chart[cIndex] do
    if next symbol ∈ NONTERMINALS then
      PREDICTOR(state, cIndex)
    elseif next symbol ∈ TERMINALS then
      SCANNER(state, cIndex)
    else
      COMPLETER(state, cIndex)
  end
while(hasNextChart)
return chart

procedure ENQUEUE(state, chart-entry)
if state not in chart-entry then
  PUSH(state, chart-entry)

procedure SCANNER((A → α • Bβ), cIndex)
if B ∈ TERMINALS then
  ENQUEUE((A → αB • β), chart[cIndex+1])

procedure PREDICTOR((A → α • Bβ), cIndex)
if B ∈ NONTERMINALS and
if no B-productions have been expanded then
  ENQUEUE((B → •αβ), chart[cIndex])
    for all production rules for B

procedure COMPLETER((B → γ•), cIndex)
REF[B] += Pointer(B → γ•)
TERMINALS += B
NONTERMINALS -= B
for each (A → α • Bβ) in chart[0 → cIndex-1] do
  ENQUEUE((A → αB • β), cIndex)
if B is a new terminal then
  for each (A → α • Bβ) in chart[cIndex] do
    SCANNER((A → α • Bβ), cIndex)
```

for an entailment, and show how the improved Earley algorithm can be used to extract all MinAs.

Example 7 *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology consisting of the axioms:*

$\alpha_1 : A \sqsubseteq P_1 \sqcap Q_1$ $\quad \alpha_4 : P_2 \sqsubseteq B$
$\alpha_2 : P_1 \sqsubseteq P_2 \sqcap Q_2$ $\quad \alpha_5 : Q_2 \sqsubseteq B$
$\alpha_3 : Q_1 \sqsubseteq P_2 \sqcap Q_2$

$\mathcal{O}$ in normal form is:

$\omega_1 : A \sqsubseteq P_1$ $\quad \omega_2 : A \sqsubseteq Q_1$ $\quad \omega_3 : P_1 \sqsubseteq P_2$
$\omega_4 : P_1 \sqsubseteq Q_2$ $\quad \omega_5 : Q_1 \sqsubseteq P_2$ $\quad \omega_6 : Q_1 \sqsubseteq Q_2$
$\omega_7 : P_2 \sqsubseteq B$ $\quad \omega_8 : Q_2 \sqsubseteq B$

Then $\mathcal{CFG}_\mathcal{O}$ for the entailment $\mathcal{O} \models A \sqsubseteq B$ is:

$\sigma_1 : S \rightarrow B$ $\quad \sigma_4 : B \rightarrow Q_2$ $\quad \sigma_7 : B \rightarrow P_2$
$\sigma_2 : Q_2 \rightarrow Q_1$ $\quad \sigma_5 : Q_2 \rightarrow P_1$ $\quad \sigma_8 : P_2 \rightarrow Q_1$
$\sigma_3 : P_2 \rightarrow P_1$ $\quad \sigma_6 : Q_1 \rightarrow A$ $\quad \sigma_9 : P_1 \rightarrow A$

The chart returned by the algorithm consist of only two chart entries for this problem as shown in Table 4. With the final completion reference list shown in Table 5. Extracting all parse trees using a depth first search results in all the MinAs being extracted for the problem as shown in Table 6.

Table 4: Solution chart

Chart 0

1.	$S \rightarrow \bullet B$	Initial State
2.	$B \rightarrow \bullet Q_2$	Predictor from 1
3.	$B \rightarrow \bullet P_2$	Predictor from 1
4.	$Q_2 \rightarrow \bullet P_1$	Predictor from 2
5.	$Q_2 \rightarrow \bullet Q_1$	Predictor from 2
6.	$P_2 \rightarrow \bullet P_1$	Predictor from 3
7.	$P_2 \rightarrow \bullet Q_1$	Predictor from 3
8.	$P_1 \rightarrow \bullet A$	Predictor from 4 and 6
9.	$Q_1 \rightarrow \bullet A$	Predictor from 5 and 7

Chart 1

10.	$P_1 \rightarrow A\bullet$	Scanner from 8
11.	$Q_1 \rightarrow A\bullet$	Scanner from 9
12.	$P_2 \rightarrow P_1\bullet$	Completer (10-6)
13.	$Q_2 \rightarrow P_1\bullet$	Completer (10-4)
14.	$Q_2 \rightarrow Q_1\bullet$	Completer (11-5)
15.	$P_2 \rightarrow Q_1\bullet$	Completer (11-7)
16.	$B \rightarrow P_2\bullet$	Completer (12-3), (15-3)
17.	$B \rightarrow Q_2\bullet$	Completer (13-2), (14-2)
18.	$S \rightarrow B\bullet$	Completer (16-1), (17-1)

Table 5: Completed reference list

REF[S]	=	[18]	REF[B]	=	[16, 17]
REF[Q_2]	=	[13, 14]	REF[P_2]	=	[12, 15]
REF[Q_1]	=	[11]	REF[P_1]	=	[10]
REF[P_1]	=	[10]			

Table 6: Extracted MinAs

$MinA_1$:	18	16	12	10
$MinA_2$:	18	16	15	11
$MinA_3$:	18	17	13	10
$MinA_4$:	18	17	14	11

Mapping back to the origional axioms we have:

$MinA_1$:	α_4	α_2	α_1
$MinA_2$:	α_4	α_3	α_1
$MinA_3$:	α_5	α_2	α_1
$MinA_4$:	α_5	α_3	α_1

7 Conclusion and future work

The combinatorial nature of MinA extraction makes it an inherently hard problem, with most approaches extracting a module based on reachability or syntactic locality. The set of axioms within this module is then systematically reduced by various methods, after which subsumption tests determine if the desired entailment still holds. Though these aproaches work well, the cost of repetitive subsumption testing is prohibitive. It is therefore desirable to eliminate as many axioms as possible before each subsumption test is performed. To this end partition methods can be employed, with the hope of eliminating large chunks of axioms that do not play a role in an entailment.

The Earley algorithm presented, based on bi-directional reachability, aims to extract all reachability based paths for an entailment directly, without first extracting smaller modules. Each parse tree extracted by the algorithm corresponds to a minimal axiom set such that reachabilty between the sub- and super-concepts in an entailment is preserved. A standard subsumption test is then performed to test

whether the axiom set is a valid MinA. This has the potential to reduce the number of subsumption tests drastically since for each parse tree the Earley algorithm extracts, the set of axioms extracted is minimal. No additional procedures need to be employed to further reduce the set of axioms and only a single subsumption test is neccesary in order to determine if the set represents a valid MinA.

We require two mapping layers, the first mapping between the original axioms in the ontology and the axioms in the normal form, the second between the normal form axioms and the production rules. Though these mappings may seem to introduce a high memory and computational overhead, in our opinion they perform an important function in debugging ontologies. Consider the axiom $A \sqsubseteq B \sqcap C$ which forms part of a MinA for some entailment, where only concept C actually plays a role in the entailment. The mappings allow us to identify exactly which concepts play a role in the entailment. Therefore instead of just presenting whole complex axioms for debugging, we have the ability to highlight exactly which concepts within the axioms are relevant to the entailment.

There are two possible problems with our approach: The first being that parse trees are minimal bi-directionally preserving axioms sets, and since they are minimal, it may occur that that all such sets are only subsets of a MinA. Thus not all MinAs may be obtained as parse trees. This boils down to the completeness question of the algorithm i.t.o. finding justifications. The second issue is that there does not exist a one-one correspondence between the axioms in the different mapping layers. Therefore when mapping back from a minimal parse tree to original axioms, we may find that the set of axioms is not a MinA anymore, in that it contains extra axioms. Though we do not directly address these issues in the current paper, we believe that the ideas presented in this paper, are both interesting and promosing, and as such warrant further investigation.

For future work we intend to implement the algorithm as a plugin for the widely used ontology editor Protégé[1] in order to test its usefulness in practise on large scale ontologies, as well as to optimize it as much as possible. If the algorithm proves useful we will investigate the possibility of extending it towards more expressive DLs. We also aim to investigate the possible link between our appraoch and automata-based pinpointing approaches (Peñaloza 2008).

References

Baader, F., Brandt, S. & Lutz, C. (2008), Pushing the $\mathcal{EL}$ envelope further., *in* K. Clark & P. F. Patel-Schneider, eds, 'OWLED 2008 DC Workshop on OWL: Experiences and Directions'.

Brandt, S. (2004), Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – what else?, *in* R. L. de Mántaras & L. Saitta, eds, 'ECAI-2004: Proceedings of the 16th European Conference on Artifcial Intelligence', IOS Press, pp. 298–302.

Chiang, Y. & Fu, K. (1984), 'Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(3), 302–314.

Earley, J. (1970), 'An efficient context-free parsing algorithm', *Communications of the Association for Computing Machinery* **13**(2), 94–102.

Grau, B. C., Horrocks, I., Kazakov, Y. & Sattler, U. (2007), Just the right amount: Extracting modules from ontologies, *in* C. Williamson & M. Zurko, eds, 'WWW '07: Proceedings of the 16th International Conference on WWW', ACM, New York NY, USA, pp. 717–726.

Grau, B. C., Horrocks, I., Kazakov, Y. & Sattler, U. (2008), 'Modular reuse of ontologies: Theory and practice', *Journal of Artifcial Intelligence Research* **31**, 273–318.

Jianfeng Du, G. Q. & Ji, Q. (2009), Goal-directed module extraction for explaining OWL DL entailments, *in* K. Thirunarayan, ed., 'ISWC'09: Proceedings of the 18th International Semantic Web Conference'. To appear.

Jurafsky, D. & Martin, J. (2009), *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, 2 edn, Prentice Hall.

Pavlatos, C., Koulouris, A. & Papakonstantinou, G. (2003), Hardware implementation of syntactic pattern recognition algorithms, *in* M. Hamza, ed., 'IASTED International Conference on Signal Processing and Pattern Analysis', Acta Press, pp. 360–365.

Peñaloza, R. (2008), Automata based pinpointing for DLs, *in* F. Baader, C. Lutz & B. Motik, eds, '21st International Workshop on Description Logics', CEUR Workshop Proceedings.

Suntisrivaraporn, B. (2009), Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies, PhD thesis, Technical University of Dresden.

[1] http://protege.stanford.edu/