CHAPTER FIVE

MINIMAL REPRESENTATION GRAPHS

5.1 INTRODUCTION

In Chapter 4 we analysed the grapheme-to-phoneme conversion task and developed an algorithm suitable for bootstrapping. During the development of this algorithm (*Default&Refine*) an interesting trend was observed: if different rule sets that all provide complete recovery of a set of training data are extracted, the smaller rule sets tend to generalise better on an unseen test set. This is not an atypical situation when addressing machine learning problems, but leads us to an interesting theoretical question: is it possible to define an algorithm that extracts the smallest possible rule set within the rewrite rule framework studied in the previous chapter, from any given set of training data? All the algorithms discussed in Chapter 4 use heuristic information to attempt to obtain such a rule set; we are interested in understanding the exact options available when attempting to obtain a minimal rule set given a set of training data.

In Section 5.2 we describe a conceptual approach that allows us to analyse the interdependencies among words in the training data in a rigorous fashion. This framework provides us both with a basis for analysing current rule extraction algorithms, and points towards a method for the extraction of a provably minimal rule set. In Section 5.3 we define the discussed framework in more detail, and demonstrate how this framework can be used to extract rule sets. In Section 5.6 we discuss the implications of our results.

5.2 CONCEPTUAL APPROACH

In this section we provide a conceptual overview of the suggested approach, referred to as *minimal representation graphs* in the remainder of this thesis. We use the same rewrite rule formalism as utilised in Chapter 4; that is, each rule describes the mapping of a single grapheme to a single phoneme using the format:

$$x_1..x_m - g - y_1..y_n \rightarrow p \tag{5.1}$$

Here g indicates the focal grapheme, x_i and y_j the graphemic context, and p the phonemic realisation of the grapheme g. The rule set is accompanied by an explicit rule application order. A pronunciation prediction for any specific word is generated one focal grapheme at a time, by applying the first matching rule found when searching through the rule set according to the rule application order. Initially we focus on a training data set that does not contain any variants, that is, every word is associated with a single unique pronunciation[1].

The goal of the approach is to obtain the smallest possible rule set that describes a set of training data completely, as an indirect approach to obtaining optimal accuracy on an unseen test set. In order to better analyse the options available when attempting to extract such a rule set, we define a framework that relies on four main observations:

1. If, for every training word, we extract all the sub-patterns of that word (as illustrated in Table 5.1), we obtain a list of all the rules that can possibly be extracted from the training data. Some of these rules will conflict with one another with regard to phonemic outcome, and we refer to these rules as *conflicted* rules. By choosing any subset of the full set of rules, and assigning a specific outcome to each rule, all possible rule sets can be generated, whether accurate in predicting the training data, or not.

Table 5.1: *The relationship between a word and its sub-pattern rules.*

Example	grapheme e to phoneme E in word 'test'
Word pattern	#t-e-st# $\rightarrow$ E
Sub-patterns	-e- $\rightarrow$ E,-e-s $\rightarrow$ E,t-e- $\rightarrow$ E,t-e-s $\rightarrow$ E t-e-st $\rightarrow$ E, #t-e-s $\rightarrow$ E,-e-st# $\rightarrow$ E #t-e-st $\rightarrow$ E,t-e-st# $\rightarrow$ E,#t-e-st# $\rightarrow$ E

2. If all the orderings among the full set of possible rules (say Z) that may be required by a subset of Z to be accurate in predicting the training data can be defined, then it becomes possible to construct a rule graph of the full rule set according to all the orderings possible, and to define appropriate operations that can manipulate this rule graph in well defined ways. During graph manipulation, specific outcomes can be assigned to rules and rules identified as *required* or superfluous. Superfluous rules can consequently be deleted, until only a minimal rule set is retained.

3. During rule prediction, the relative rule application order of two rules that occur in an extracted rule set is only of importance if the two rules conflict with regard to outcome, and if both can

[1]We discuss options for dealing with pronunciation variants in Section 5.5

apply to a single word. During rule extraction, the order in which two rules occur in an interim rule set is only of importance if both can apply to a single word in the training data, and that word has not yet been ‘caught’ by any required rule occurring earlier in the rule set. For each rule, we refer to the latter set of words as the *possible words* associated with that rule.

4. The full set of possible rules Z cannot occur in any order. It is possible to restrict the allowable orderings between any two rules for two reasons: (1) if one rule is more specific than another, the first rule must occur earlier in the rule set than the second in any minimal rule set. If not, the second (more general) rule will always be invoked when predicting a word that applies to both rules, and the first rule will be redundant (which is impossible if the rule set is minimal); and (2) if two rules are applicable to the same word in the training data but conflict with regard to outcome. For such rules the words shared in the *possible words* sets of each rule dictate the orderings that are valid.

Using the above observations, we can analyse a set of training data in order to understand the interdependencies among words in the training data, and the options for extracting a minimal rule set. We illustrate the concepts using a simple 3-word example, consisting of the words ‘test’,‘ten’ and ’tea’ and consider the steps required to extract a rule set for the letter ‘e’. As the software that we developed to implement this approach uses a single character representation of each grapheme and phoneme, we do the same in this example.

Prior to rule extraction, a *word pattern* is generated from each aligned word-pronunciation pair in the training data, as shown in Table 5.2. Hashes denote word boundaries.

Table 5.2: *Word patterns associated with the words ‘test’,‘ten’ and ‘tea’.*

	aligned ARPAbet example	single character representation
Words	t e s t $\rightarrow$ t eh s t	t e s t $\rightarrow$ t e s t
	t e n $\rightarrow$ t eh n	t e n $\rightarrow$ t e n
	t e a $\rightarrow$ t iy ϕ	t e a $\rightarrow$ t i ϕ
Word patterns	#t-e-st# $\rightarrow$ eh	#t-e-st# $\rightarrow$ e
	#t-e-n# $\rightarrow$ eh	#t-e-n# $\rightarrow$ e
	#t-e-a# $\rightarrow$ iy	#t-e-a# $\rightarrow$ i

For each of the word patterns, we generate a set of sub-patterns (as listed in Table 5.1 for the word pattern #t-e-st# $\rightarrow$ e). These sub-patterns are arranged in a graph structure according to specificity, with the more general rules later in the graph (closer to the root), and more specialised rule earlier (higher up in the graph). Initially, an ordering is only added between two rules where the context of one rule contains the context of another, and we refer to these orderings as *contain pattern* relationships. A topological sort of this graph will result in a rule set that is accurate, but contains a large number of superfluous rules. From the outset, the process assumes that any of the rules may be deleted in future. As it becomes clear that certain rules are required in order to retain accuracy over the training data (irrespective of further allowed changes to the rule set), these rules are marked as *required* rules.

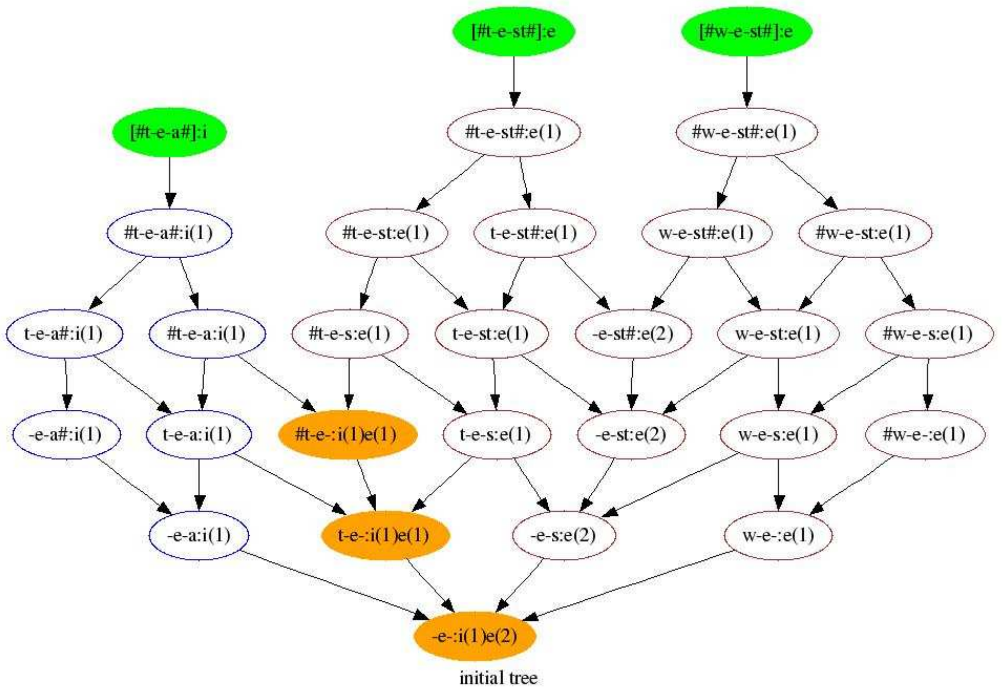

Figure 5.1: *An example rule graph, corresponding to the word patterns in Table 5.2*

This process is illustrated in Fig. 5.1. Word nodes (one per word pattern) are indicated in green. Clear nodes indicate rule nodes that can only predict a single outcome. For these nodes, different coloured outlines indicate different outcomes. Orange nodes are associated with more than one possible outcome: different choices with regard to outcome will result in different rule sets. Black edges indicate that an ordering between two rules is required, irrespective of further rule graph manipulation. In the initial graph these edges represent *contain pattern* relationships. Currently no rules are marked as required; if there were, these would be marked in yellow.

Orderings are transitive. If all the orderings implied by the current set of edges are considered, then the only additional orderings that can possibly occur in the full rule set are between rules that share a word in their respective *possible words* set, and have not already been assigned a fixed ordering. We refer to these rules as *minimal complements*. These *minimal complement* relationships are added and utilised during rule extraction. We do not indicate them explicitly on all the graphs used to illustrate the current example, as the addition of minimal complement relationships results in visually complex graphs. For illustration, we mark the minimal complements related to a single rule '-e-st' for the initial graph of Fig. 5.1 and display the result in Fig. 5.2. Minimal complement relationships are marked as orange edges.

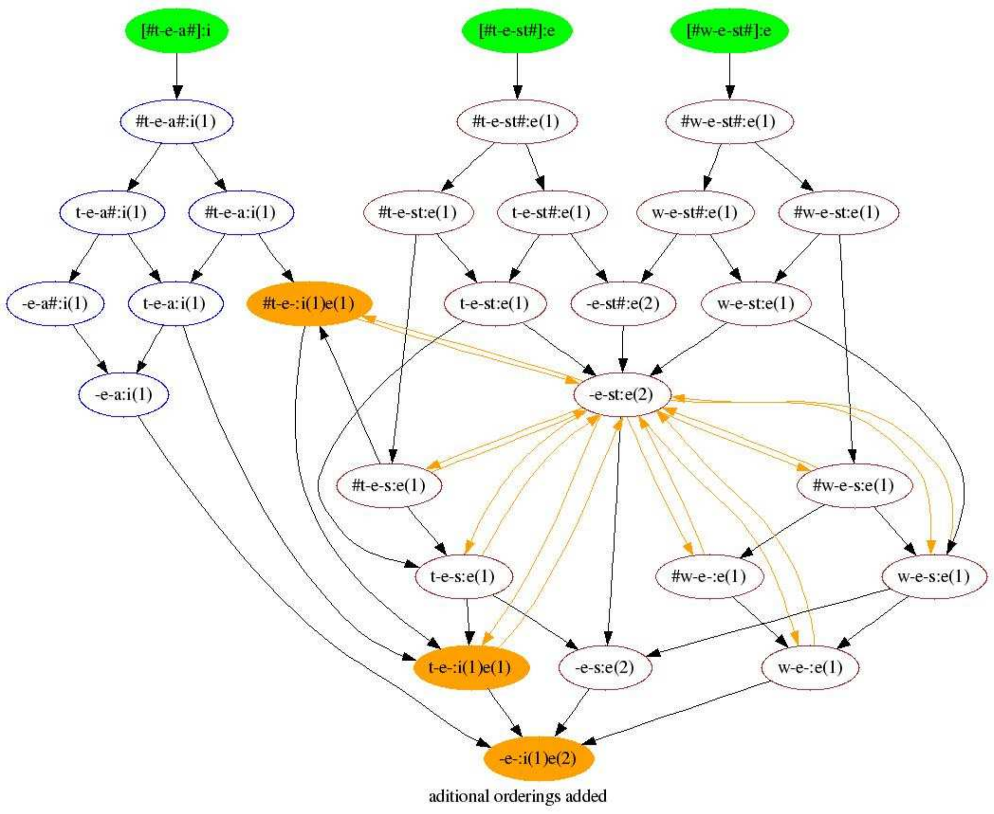

Figure 5.2: *Marking the minimal complement relationships associated with the rule '-e-st' for the rule graph of Fig. 5.1.*

Note that the minimal complements associated with any rule r can only occur in a restricted range: the context of the earliest rule may not contain rule r, and the context of the latest rule may not be contained by r itself. As this range is restricted, the number of additional orderings that may be required is similarly restricted. Each additional minimal complement pair added to the graph introduces two possible orderings. This increases the number of options to consider when making any single decision (whether to resolve a conflicted node to a single outcome, or whether a specific rule is required or can be deleted.) We would like to remove as many of the 'double orderings' as possible, and replace these with orderings that indicate a single direction. In some cases additional information is available to choose one of the orderings and discard the other:

- If the possible words associated with a rule r is a subset of the possible words of a second rule s, rule r must always occur earlier in the rule extraction order than s. The reasoning is similar to that followed when adding the initial contain pattern orderings, but now holds for minimal complements that are not necessarily in a contain pattern relationship. We refer to these relationships as *super complements*. While contain pattern relationships can be added to the graph from the outset, *super complements* emerge as the rule set extraction process progresses. As more rules are marked as $required$, the possible words sets of later rules decrease, and super complement relationships start to emerge. Once an ordering is added between two super complements, this relationship is not changed at a later stage during rule manipulation[2]

- If a rule r predicts a single outcome, and accurately matches all the words in the intersection of the possible words of rule r and the possible words of another rule s, and there is at least one word in this set that s will mispredict given any of its allowed outcomes, then rule r has to occur before rule s for the rule set to be accurate. We refer to these relationships as *order required* relationships. If neither of the two rules matches the full set of shared words, the relationship is still inconclusive. As with super complement relationships, order required relationships also emerge as the rule set extraction process progresses.

In Fig. 5.3 we identify and add additional super complement relationships. The current rule graph does not have any order required relationships among nodes.

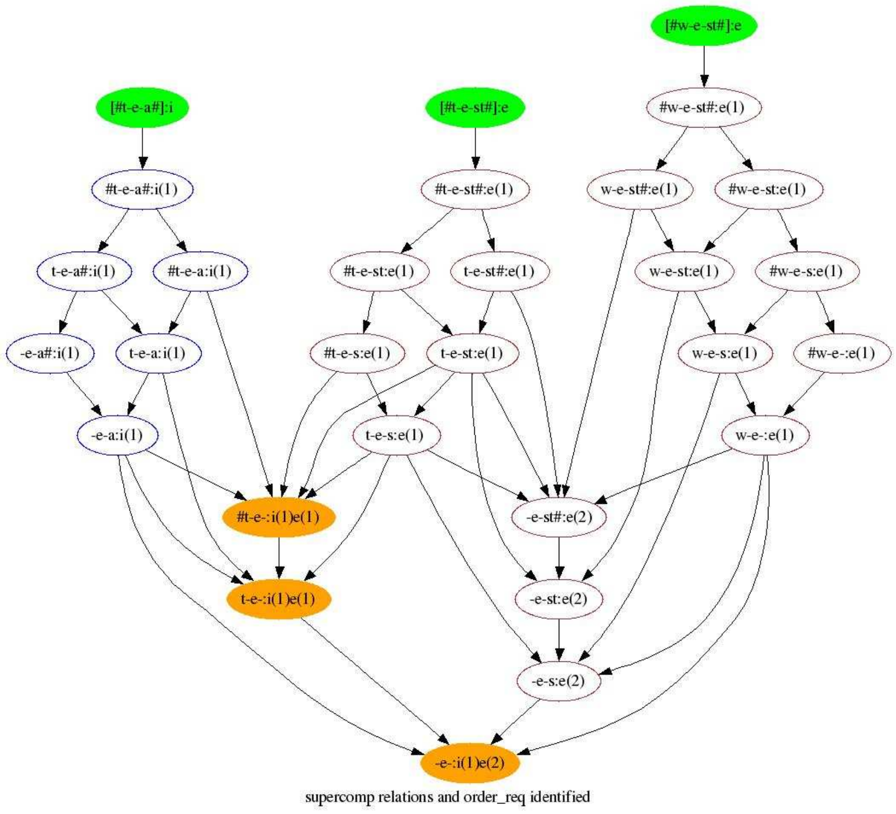

Figure 5.3: *Adding super complements to the rule graph of Fig. 5.1. (Minimal complements are not shown.)*

Since orderings are transitive, we can remove any definite orderings that are already implied by others. For example, in Fig. 5.3 the relationship between rules 't-e-st' and '#t-e- is already implied

[2]As more rules are marked as required, the possible words sets of all other rules become smaller. If a set of possible words associated with a rule r is the subset of the possible words associated with a rule s, this relationship will be maintained unless both sets become equal. In the latter case, one of the two rules are redundant and will be deleted during rule extraction, as discussed later. Since either r or s will be deleted, the ordering between these two rules become insignificant, and the prior ordering based on their previous super complement relationship may be retained without restricting rule graph manipulation options.

by the relationships between rules ‘t-e-st’ and ‘t-e-s’, and between rules ‘t-e-s’ and ‘#t-e-’. Such redundant edges can be removed without losing any information currently captured in the rule graph.

This process is illustrated in Fig. 5.4. Note how the relationships become simpler and the graph more loosely connected from Fig. 5.1 to Fig. 5.4.

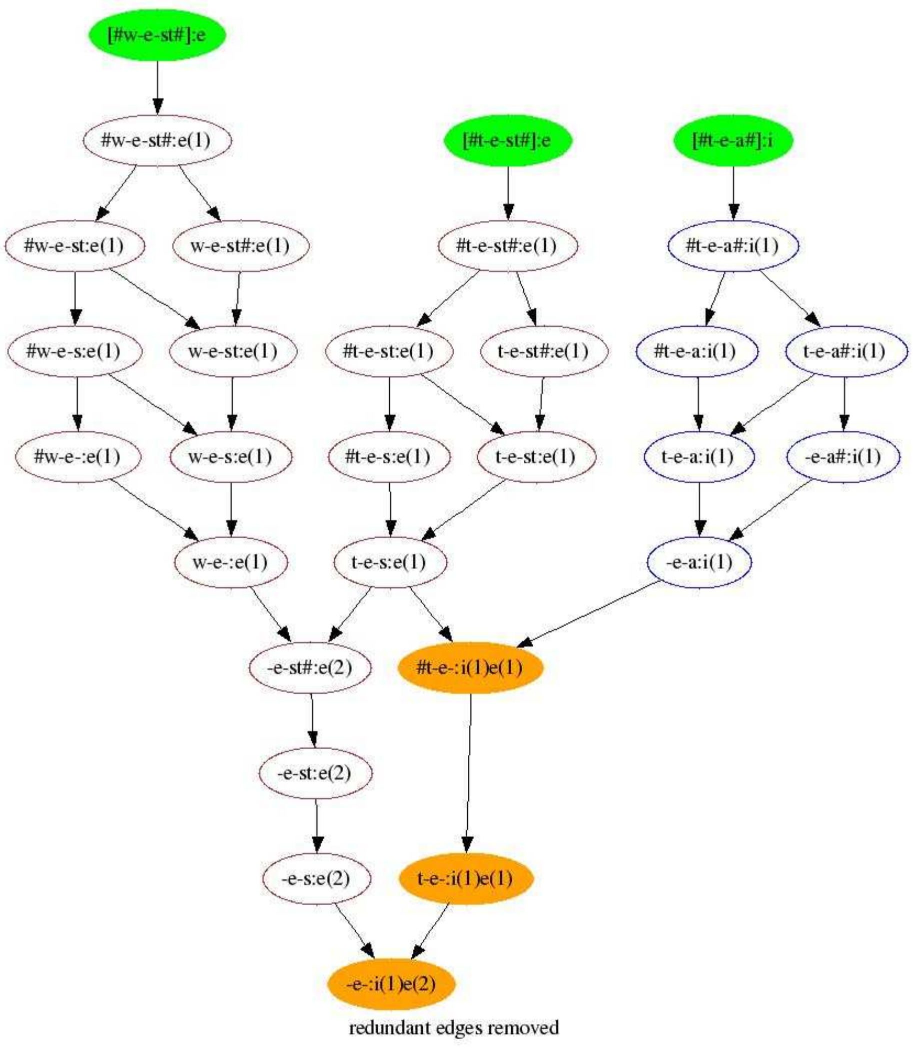

Figure 5.4: *Removing unnecessary edges from the rule graph of Fig. 5.3. (Minimal complements are not shown.)*

If we can be sure that we have added all the necessary orderings (caused by contain pattern, super complement or order required relationships) and we keep track of all minimal complement relationships that still have an uncertain ordering, we now have a rule graph that both contains all possible rules, and specifies all possible orderings that may be required to define a valid rule application order.

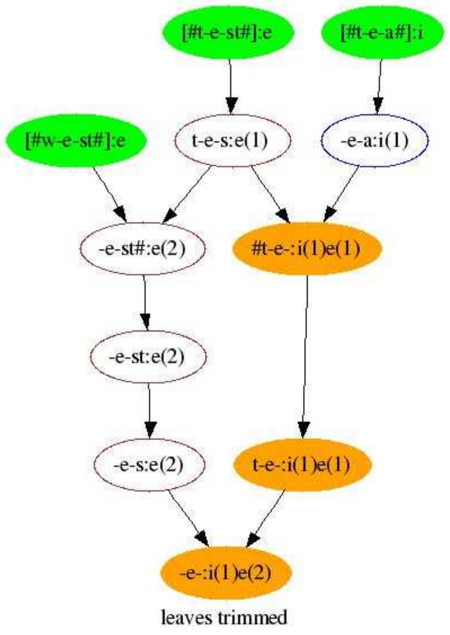

Figure 5.5: *Removing unnecessary rules from the rule graph of Fig. 5.4.(Minimal complements are not shown.)*

We can now use this rule graph as basis to make decisions about which outcome to select where a rule is conflicted (has more than one outcome), or even decide when a rule can be deleted or not.

When rules are deleted, it is possible that one of the rules required by a minimal rule set is deleted unintentionally, and in order to compensate for this deletion, two or more additional rules may have to be kept to retain accuracy over the training data. The final rule set will then have more rules than strictly required. To prevent this from happening, rules are eliminated by deleting redundant rules, identifying required rules and resolving conflict rules via a small set of allowed operations. The $state$ of rule extraction can always be described by a triple consisting of the possible rules that can be included in the rule set (Z'), the rules that have been marked as required (Z_e), and the orderings that are definite ($oset(Z')$, the black edges in the graph). Additional orderings that are possible can automatically be generated from such a state. Each allowed operation changes the state of rule extraction, from one *allowed state* to another, with the initial allowed state as depicted in Fig. 5.1.

One example of such an allowed deletion operation can be illustrated as follows: The rule graph in Fig. 5.4 clearly contains a number of superfluous rules. Whenever a rule r exists such that (1) it is not conflicted, and (2) all the possible words associated with rule r can be caught by one or more immediate successors that agree with rule r with regard to outcome, and (3) rule r does not have any immediate successors that can potentially disagree with regard to outcome, then rule r can safely be deleted from the rule graph. All rules that meet these conditions, can be deleted from the rule graph, as illustrated in Fig. 5.5. Since the rule graph is now significantly simpler, we start displaying the remaining minimal complements from Fig. 5.6 onwards.

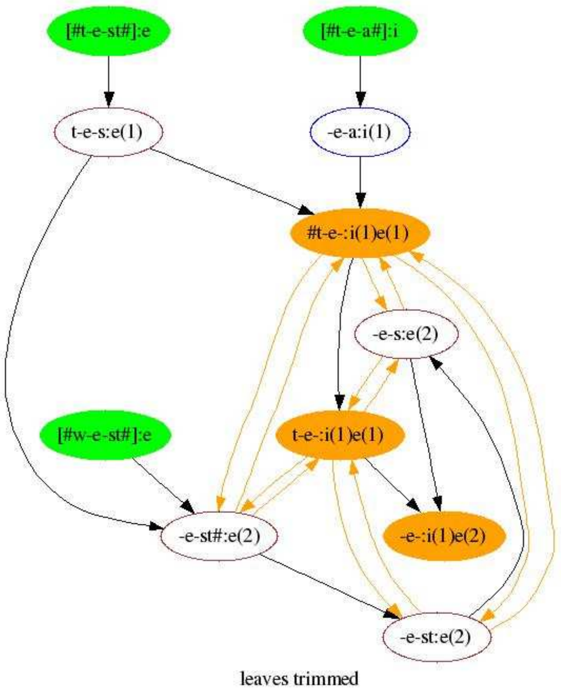

Figure 5.6: *Removing unnecessary rules from the rule graph of Fig. 5.4.(Minimal complements are shown.)*

Where the possible words associated with rule r are exactly the same as the possible words of any one of its successors s, rule r and rule s are deemed *rule variants*. Either of two rule variants can be generated at the same point in the rule extraction order, without influencing the number of rules in the final rule set. The process keeps track of all deleted rules that are variants of retained rules. In this way, while a rule node is physically deleted, the rules are in effect merged, and either of the two rules may be utilised in the final rule set, as discussed later.

Additional deletion operations identify rules that have an empty set of possible words, and rules that are true variants of another, that is, two rules that are both resolved to a single outcome, and have identical relationships with identical predecessors and successors. While these deletion operations create a rule graph that is significantly simpler, we have not yet made any decisions with regard to the best choice of outcome for any of the conflicted nodes. Prior to rule resolution, we first identify any rule r as *single* where – given the current state of rule extraction – at least one word can only be predicted by either rule r or by another rule directly in the path of r. In the remaining figures, these single rules are marked '*S'.

There are various conditions under which a conflicted rule can be resolved, one of which we illustrate here. Conflicted nodes can be thought of as 'default' or 'fallback' nodes. During pronunciation prediction, a fallback node will only be invoked if a more specialised rule is not available that matches the word being predicted. These nodes therefore only need to be retained if, in some way or another, the rule can generalise from its immediate predecessors. This requires that at least two predecessors should predict a similar outcome. If this is not the case, the fallback node does not provide any further advantage, and can be removed from the rule graph without constraining the rule set in a way that does not allow final minimisation[3]. This process is illustrated in Fig. 5.7 and Fig. 5.8.

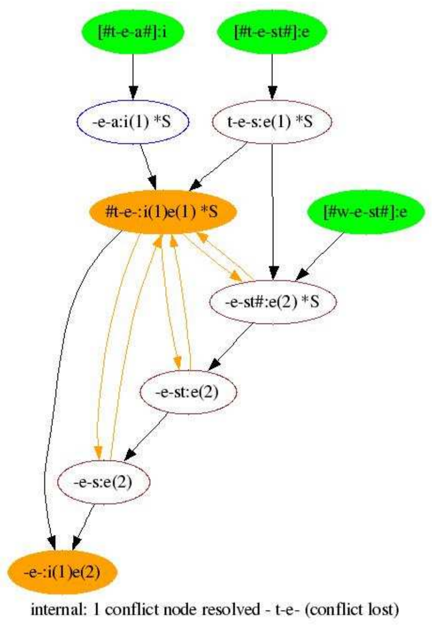

Figure 5.7: *Resolving conflicted rule 't-e-'.*

[3]This does not apply to the root node. The root node is handled as a special case, as discussed below.

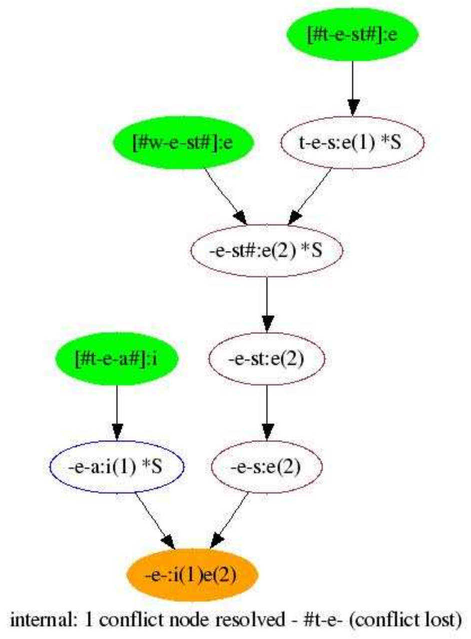

Figure 5.8: *Resolving conflicted rule '#t-e-'.*

Note that in the Fig. 5.8, none of the minimal complement relationships have been retained. Additional resolution operations analyse the definite and possible predecessors and select a specific outcome based on this analysis. When resolving a conflicted rule to a specific outcome, it is required that at least one of the predecessors that has an outcome that matches the outcome selected for resolution must be marked as a *single* rule. If such a single rule exists, this implies that some rule with the selected outcome will be generated at this point in the rule extraction order. While there is not certainty that such a rule is required, the conflicted rule may not yet be resolved.

Applying the same deletion operator discussed earlier, three additional rule nodes can be deleted, as illustrated in Fig. 5.9.

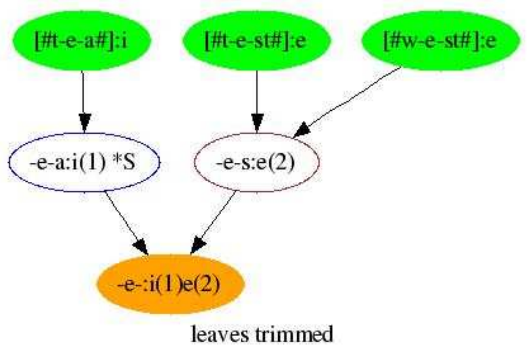

Figure 5.9: *Removing unnecessary rules '-e-st', '-e-st#' and 't-e-s'.*

If the resolution operator discussed previously were to be applied to the root node, the rule set would remain valid. However, this would result in the root node being deleted, and it is easier in practise to manipulate the graph assuming a single root node. Also, we would like to generate some 'default rule' that can be used to predict any word pattern not previously seen. Therefore the root node is always resolved to a single outcome, once all its predecessors are resolved (and not deleted, as would be the case if the standard resolution operator were applied). Resolving the root node to a single outcome when standard application of a deletion operator indicated that it should have been deleted, is similar to choosing one variant of a rule above another variant of the same rule. As all variants are retained during rule extraction, and the final choice with regard to which variant to choose is postponed until after graph minimisation, manipulating the root node as a special case does not restrict the rule extraction process in any way. In Fig. 5.10 the root node is resolved to one of its possible outcomes.

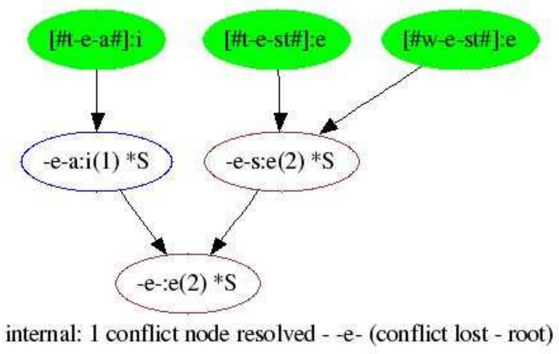

Figure 5.10: *Resolving conflicted rule '-e-'.*

If for at least one word pattern w in the possible words set of a rule r, there exists no other rule than can possibly predict word pattern w correctly, given the current state of rule extraction (the remaining rule set, the required rule set and the decided orderings); then rule r is a $required$ rule and can be marked as such. When a rule is identified as a required rule, all words in the possible words set of rule r are removed from the possible words sets of rules occurring later in the rule graph. In Fig 5.11 two rules are marked as required, with required nodes indicated in yellow. One final deletion (using the standard deletion operator) and the minimal rule set is obtained, as depicted in Fig. 5.12.

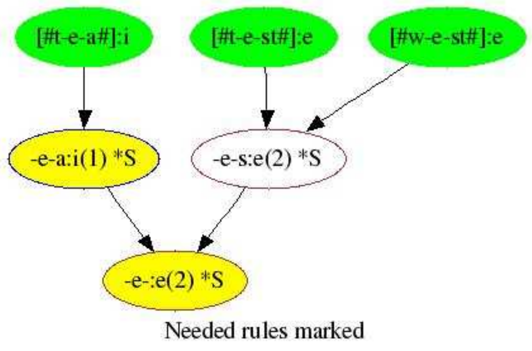

Figure 5.11: *Identifying required rules '-e-a' and '-e-'.*

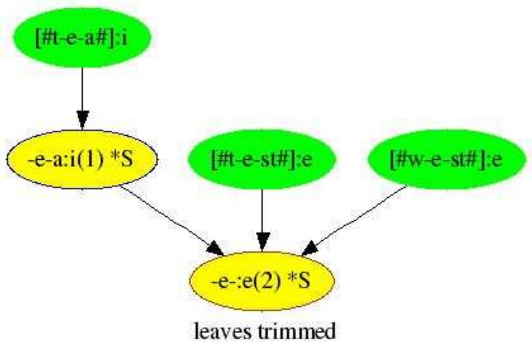

Figure 5.12: *The final (minimal) rule graph.*

The rule set that can now be extracted from the rule graph by performing a topological graph traversal. This results in the rule set listed in Table 5.3. For each extracted rule, a number of possible variants are listed. A rule can be replaced by any of its variants without affecting the accuracy of the rule set, or requiring the inclusion of additional rules. Note that for any single word that gives rise to a single rule (such as the word pattern #t-e-a# in this example), all word sub-patterns that have not been identified as currently part of the rule set are included as variants.

Table 5.3: *The final rule set generated from the words in Table 5.2, including possible variants.*

Rule number	Extracted rule	Possible variants
1	-e-a $\longrightarrow$ i	#t-e-a #t-e-a# -e-a# t-e-a# t-e-a
2	-e- $\longrightarrow$ e	-e-st# -e-s -e-st

At this stage, heuristic choices related to characteristics such as rule context size, rule context symmetry, or variance with regard to the training data can be utilised to choose the most appropriate rule set. In larger rule sets, many rules do not have variants, but a relatively large proportion of rules retain one or more variants. The ability to make heuristic choices late in the rule extraction process, provides significant flexibility in obtaining the appropriate rule set.

5.3 THEORETICAL FRAMEWORK

In this section we describe the above framework in more detail, and provide a more rigorous definition of the terminology used[4]. We provide proofs for the key statements in Appendix B. When we refer to a specific statement in the text, we are referring to the statement as found in Appendix B.

Firstly, we define the rule format and the various terms used during rule set analysis. We then proceed to show how a relationship between two rules in a minimal rule set translates to a specific relationship between the same two rules in a larger rule set, and describe the conditions and implications of a rule ordering between two rules occurring in either of these types of rule sets. Using these conditions, we provide a formal definition of an allowed state of rule extraction. We analyse the characteristics of an allowed state and define an initial state that can be shown to be allowable in these terms. We then define the various allowed operations that, when applied, progress the rule graph from one allowed state to another. In contrast to the overall framework, the set of allowed operations are still somewhat experimental, as discussed in section 5.3.6. Finally, we discuss the minimality of the extracted rule set and describe additional options for the improvement of generalisation ability.

5.3.1 RULE FORMAT

As discussed in Section 5.2, we use a set of rewrite rules to describe the mapping of a single grapheme to a single phoneme.

If G is the set of possible graphemes and H the set of possible phonemes; the i^{th} rule for grapheme g is formulated as

$$\begin{aligned} rule(g,i) &= (x_1..x_m, g, y_1..y_n) \rightarrow z; \\ x_1..x_m, g, y_1..y_n &\in G; z \in H; \end{aligned} \tag{5.2}$$

alternatively written as:

$$x_1..x_m - g - y_1..y_n \rightarrow z$$

where $x_1..x_m$ defines the m-grapheme left context of g, $y_1..y_n$ defines the n-grapheme right context of g, and z is the predicted phonemic realisation of grapheme g when found within the given left and right word contexts. G includes ϕ_G, the null grapheme and $\#$ the word boundary marker (with always $g \neq \#$). H includes ϕ_H, the null phoneme.

[4]Terms and definitions are presented in definition boxes, interspersed among more general comments.

The $outcome(r)$ function describes the phonemic outcome of the rule r:

$$outcome(rule(g,i)) = outcome(x_1..x_m - g - y_1..y_n \rightarrow z) = z. \tag{5.3}$$

The $context(r)$ function describes the application context of the rule[a] r directly:

$$context(rule(g,i)) = context(x_1..x_m - g - y_1..y_n \rightarrow z) = x_1..x_m - g - y_1..y_n. \tag{5.4}$$

[a] $context(.)$ can also be applied to word patterns, as defined in eq. 5.9

The rule application order $rule\_order(Z', r, s)$ specifies the order in which any two rules r and s occurring in a rule set Z' are applied, where

$$\forall r, s \in Z' : rule\_order(Z', r, s) = 1 \implies rulenum(r) < rulenum(s) \tag{5.5}$$

and the $rulenum(r)$ function describes the rule number of a specific rule r directly:

$$rulenum(rule(g,i)) = i. \tag{5.6}$$

The $oset(Z')$ for a rule set Z' consists of the entire set of orderings specified by $rule\_order(.)$, i.e:

$$oset(Z') = closure(Z', rule\_order(.)) \tag{5.7}$$

where $rule\_order(.)$ defines the current set of orderings over Z' and $closure(.)$ consists of the transitive closure of the set of rule pairs for which a specific relation is defined, i.e.:

$$\begin{aligned} closure(Z', relation(.)) = \cup_i (r,s) \forall r, s \in Z' : \\ relation(Z', r, s) = 1 \text{ or } \exists t \in Z' : relation(Z', r, t) = 1, \\ relation(Z', t, s) = 1. \end{aligned} \tag{5.8}$$

The $rule\_order(.)$ relation restricts the $rulenum(.)$ function to a set of options, and does not necessarily specify an ordering between every two rules. If rules are applied according to the $rule\_order(.)$ relation and an ordering between two rules that both match a word is indeterminate, either of the rules can potentially be invoked. It is possible to convert from an implicit $rule\_order(.)$ to an explicit rule numbering via the $assign(.)$ function:

Let the set $assign(Z', oset(Z'))$ define all the possible rule number assignments that are valid given the specified rule set Z' and rule orderings $oset(Z')$. Per assignment, a single rule number is assigned to every rule, consistent with $oset(Z')$.

Note that for a specific value of $assign(Z', oset(Z'))$, $rulenum(r) < rulenum(s)$ does not imply that $rule\_order(Z', r, s) = 1$.

A word w consists of a sequence of graphemes in G. During pronunciation prediction of a word of length n (also counting word boundaries), we create n word patterns that each focus on a specific grapheme in the word. When focusing on grapheme i, the word pattern is described as:

$$\forall w = x_1..x_n; x_j \in G\,; n \geq 1; 1 \leq i \leq n:$$

$$word\_pattern(w, i) = x_1..x_{i-1} - x_i - x_{i+1}..x_n. \tag{5.9}$$

(5.10)

The $context(w)$ function can also be applied to word patterns, where the $context$ of a word pattern w is simply the word pattern itself.

The $match(w, r)$ function indicates that a rule r occurring in a rule set Z' can be applied to predict a word pattern w:

$$\forall r \in Z' : match(w, r) = 1 \iff context(w) \supseteq context(r). \tag{5.11}$$

The $winningrule(w, g)$ relation describes the first matching rule(s) found in rule set Z' for word pattern w with regard to grapheme g, i.e

$$\forall r \in Z' : r \in winningrule(Z', oset(Z'), w, g) \iff match(w, r) = 1,$$
$$\nexists s : match(w, s) = 1, (s, r) \in oset(Z'). \tag{5.12}$$

Rules with equivalent contexts and different outcomes are not allowed in the final rule set, i.e:

$$\forall g \in G, i, j \in N : context(rule(g,i)) = context(rule(g,j)) \implies outcome(rule(g,i)) = outcome(rule(g,j)). \quad (5.13)$$

Conflicting rules will however exist during the interim steps of rule extraction, as discussed below.

5.3.2 RULE SET ANALYSIS

5.3.2.1 TRAINING DATA, WORD PATTERNS AND SUB-PATTERNS

The rule set is derived from a set of training data. As in the previous chapters, a data set consisting of aligned word-pronunciation pairs is used as input during rule extraction. Word patterns and word sub-patterns are extracted from this set, and form the basis for further rule set construction.

Each word-pronunciation pair consists of two sequences $x_1..x_n$ and $y_1..y_n$, where $n \geq 1, x_i \in G$ and $y_i \in H$. Let $TD(g)$ be the set of all word patterns in the training data that describe a specific grapheme g, associated with a specific phonemic outcome per word pattern. Then:

$$\forall g \in G : w \in TD(g) \iff w = x_1..x_{i-1} - g - x_{i+1}..x_n \rightarrow y_i, \text{ where } x_1..x_n \text{ and } y_1..y_n \text{ an aligned word-pronunciation pair.} \quad (5.14)$$

In the remainder of this section, assume g to simplify notation (for example let $rule(i)$ be equivalent to $rule(g,i)$ for the specific g being considered). TD does not contain word variants (multiple pronunciations of a single word), that is:

$$\nexists w_1, w_2 \in TD : context(w_1) = context(w_2) \implies outcome(w_1) \neq outcome(w_2). \quad (5.15)$$

A word pattern is in effect the largest possible rule that describes the grapheme-to-phoneme mapping accurately. The combined left and right contexts of the word pattern therefore contains the full word, including word boundaries. For each word pattern, a set of sub-pattern rules – describing all possible sub-contexts of the word pattern – can be generated, as previously shown in Table 5.1.

Let Z be the set of all possible word patterns and sub-patterns associated with the word patterns in TD.

For any two rule sets, Z_A and Z_B, let $Z_A \subseteq Z_B$ indicate that one set is equal to or a subset of the other, both with regard to the context and outcome of rules. More specifically:

$$Z_A \subseteq Z_B \iff r \in Z_A \implies r' \in Z_B,$$
$$context(r) = context(r'), outcome(r) \subseteq outcome(r'). \tag{5.16}$$

Let $|Z'|$ indicate the number of rules in any rule set Z', where $Z' \subseteq Z$.

Let $allset(Z')$ consist of all possible orderings in a rule set Z', whether contradictory or not:

$$\forall Z' \subseteq Z_{combined} : allset(Z') = \cup_{i,j}(v_i, v_j) \forall v_i, v_j \in Z', i \neq j. \tag{5.17}$$

A word pattern w can be referred to either as a word pattern $w \in TD$ or as a rule $w \in Z$. The set Z then consists of all possible rules that can potentially apply to the word patterns in TD.

5.3.2.2 CONFLICT RULES AND CONFLICT RESOLUTION

As the set Z consists of all possible rules that can potentially apply to the word patterns in TD, it may include a number of conflicting rules. Under certain conditions, these rules can be resolved to a specific outcome. Until a rule is resolved to a single outcome, a set of allowable outcomes is retained per rule.

Let $Z_{conflict}$ consist of all the conflicting rules in Z, that is, rules that contradict eq. 5.13. Let $Z_{no-conflict}$ be the set of remaining rules, when all conflicting rules in $Z_{conflict}$ are removed from Z, i.e

$$\begin{aligned} Z_{conflict} \cup Z_{no-conflict} &= Z. \\ Z_{conflict} \cap Z_{no-conflict} &= \phi. \end{aligned} \tag{5.18}$$

Define the $conflictrule(r_{\alpha 1}, r_{\alpha 2}, .., r_{\alpha n})$ for all n rules $r_{\alpha i} \in Z_{conflict}$ with equivalent contexts as one rule with one of n alternative outcomes, i.e:

$$\begin{aligned} \forall r_{\alpha i} \in Z_{conflict}, context(r_{\alpha i}) = context(r_{\alpha}) \forall i = 1...n : \\ conflictrule(r_{\alpha 1}, r_{\alpha 2}, .., r_{\alpha n}) = context(r_{\alpha}) \rightarrow z_1 || z_2 || ... || z_n, \\ z_j = outcome(r_{\alpha j}) \forall j = 1...n, \\ \text{where } z_j || z_k \text{ indicate that either } z_j \text{ or } z_k \text{ is a possible } outcome. \end{aligned} \tag{5.19}$$

Define the resolution of a $conflictrule$ as a specific-outcome version of the rule, i.e let:

$$\begin{aligned} \forall r_{\alpha} \in Z_{conflict}, z_x \in \cup_{r_{\alpha i}} outcome(rule(r_{\alpha i})) : \\ resolve(conflictrule(r_{\alpha 1}, r_{\alpha 2}, .., r_{\alpha n}), z_x) = context(r_{\alpha}) \rightarrow z_x. \end{aligned} \tag{5.20}$$

If a rule r with the same context is referred to with regard to different rule sets in which different resolved versions of the rule may occur, let $outcome(r|Z')$ indicate $outcome(r)$ where $r \in Z'$.

For each subset of all elements in $Z_{conflict}$ with equivalent contexts, it is possible to generate a single $conflictrule$. Let the set $Z_{conflict-combined}$ consist of all the conflict rules generated from $Z_{conflict}$ according to eq. 5.19, which have not been resolved. Let the set $Z_{conflict-resolved}$ consist of all the resolved conflict rules, where a conflict rule will move from $Z_{conflict-combined}$ to $Z_{conflict-resolved}$ upon resolution (according to eq. 5.20). Let $Z_{combined}$ consist of all elements in $Z_{no-conflict}$ combined with the elements in $Z_{conflict-combined}$ and $Z_{conflict-resolved}$, where

$$\begin{aligned}
&Z_{no-conflict} \cup Z_{conflict-resolved} \cup Z_{conflict-combined} = Z_{combined} \\
&Z_{conflict-combined} \cap Z_{no-conflict} = \phi \\
&Z_{conflict-resolved} \cap Z_{conflict-combined} = \phi \\
&Z_{conflict-resolved} \cap Z_{no-conflict} = \phi \qquad (5.21)
\end{aligned}$$

and let
$$Z_{single} = Z_{conflict-resolved} \cup Z_{no-conflict} \qquad (5.22)$$

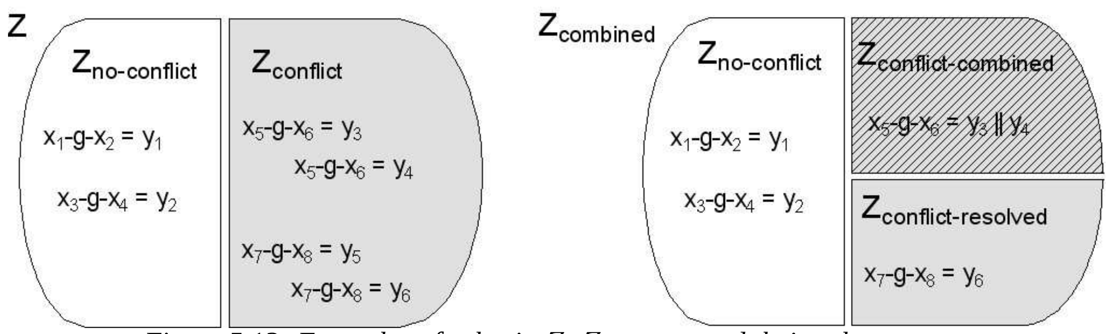

Figure 5.13: *Examples of rules in Z, $Z_{combined}$ and their subsets.*

The relationships among the different sets are depicted in Fig. 5.13.

5.3.2.3 *COMPLETE, ACCURATE, MINIMAL AND POSSIBLY_MINIMAL RULE SETS*

Any subset of $Z_{combined}$, ordered according to a specific rule ordering $rule\_order(.)$ will describe the training data with a certain degree of accuracy. The ideal rule set will be one that is not only complete but also accurate, and not only accurate but also minimal, as defined below:

A $complete$ rule set can predict all the words in the training data:

$$\begin{aligned}\forall Z' \subseteq Z_{combined} : complete(Z') = 1 \iff \\ \forall w \in TD, \exists r \in Z' : match(w,r) = 1.\end{aligned} \tag{5.23}$$

An $accurate$ rule set predicts all words in the training data accurately:

$$\begin{aligned}\forall Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ accurate(Z', oset(Z')) = 1 \iff complete(Z') = 1, \\ \forall w \in TD, \forall r \in winningrule(Z', oset(Z'), w) : outcome(w) = outcome(r).\end{aligned} \tag{5.24}$$

A $minimal$ rule set is an accurate rule set that contains the fewest rules possible:

$$\begin{aligned}\forall Z' \subseteq Z_{combined}, oset(Z') \subseteq allset(Z') : \\ minimal(Z', oset(Z')) = 1 \iff accurate(Z', oset(Z')) = 1, \\ \nexists Z'' \subseteq Z_{combined}, oset(Z'') \subseteq allset(Z'') : \\ accurate(Z'', oset(Z'')) = 1, |Z''| < |Z'|.\end{aligned} \tag{5.25}$$

A $possibly\_minimal$ rule set is a set of rules that can be minimal, if ordered correctly:

$$\begin{aligned}\forall Z' \subseteq Z_{combined} : possibly\_minimal(Z') = 1 \iff \\ \exists oset(Z') \subseteq allset(Z') : minimal(Z', oset(Z')) = 1.\end{aligned} \tag{5.26}$$

5.3.2.4 ALLOWED STATES AND ALLOWED OPERATIONS

The full set of rules in $Z_{combined}$ consists of all possible rules and is therefore a superset of one or more $minimal$ rule sets Z_m[5]. We would like to delete the unnecessary rules until only a $minimal$ rule set is retained. When rules are deleted, it is possible that one of the rules required by Z_m is deleted unintentionally, and in order to compensate for this deletion, two or more additional rules may have to be kept to retain accuracy over TD. The final rule set Z' will then have a number of rules $|Z'| > |Z_m|$. To prevent this from happening, rules are eliminated by adding orderings, deleting redundant rules, identifying required rules and resolving conflict rules via a set of allowed operations. The $state$ of rule extraction can always be described by the triple $Z', Z_e, oset(Z')$, where Z' indicates the possible rules that can still be included in the final rule set, Z_e indicates required rules that have to be included in the final rule set, and $oset(Z')$ identifies some of the required rule orderings among elements of Z'. Each allowed operation changes the $state$ of rule extraction, from one $allowed\_state$ to another, with $allowed\_state$ as defined below (in eq 5.29).

Let the $order\_subset(oset_A(Z_A), oset_B(Z_B))$ relation be true if a set of rule orderings $oset_A(Z_A)$ is equal to or a subset of another set of rule orderings $oset_B(Z_B)$ (possibly defined on a different rule set) when the two sets of rule orderings are compared on their rule set intersection. More specifically:

$$\begin{aligned}
\forall Z_A \subseteq Z_{combined}, Z_B \subseteq Z_{combined}, \\
\forall oset_A(Z_A) \subseteq allset(Z_A), oset_B(Z_B) \subseteq allset(Z_B): \\
order\_subset(oset_A(Z_A), oset_B(Z_B)) = 1 \iff \\
\forall r, s \in Z_A \cap Z_B : (r, s) \in oset(Z_A) \implies (r, s) \in oset(Z_B).
\end{aligned} \tag{5.27}$$

Let $minrules(Z', Z_e, oset(Z'))$ identify all the $minimal$ rule set and rule ordering set pairs that can be derived from Z', given the set of orderings $oset(Z')$ and a required rule subset Z_e. More specifically:

$$\begin{aligned}
\forall Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z'): \\
(Z_m, oset_m(Z_m)) \in minrules(Z', Z_e, oset(Z')) \iff \\
minimal(Z_m, oset_m(Z_m)) = 1, Z_e \subseteq Z_m \subseteq Z', \\
order\_subset(oset(Z'), oset_m(Z_m)) = 1.
\end{aligned} \tag{5.28}$$

[5]By definition, at least one $miminal$ rule set will always exist.

Let $allowed\_state(Z', Z_e, oset(Z'))$ indicate that for a given required subset Z_e and required set of orderings $oset(Z')$, there exists a $minimal$ rule set Z_m contained within Z':

$$\begin{aligned}\forall Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') :\\ allowed\_state(Z', Z_e, oset(Z')) = 1 \iff\\ \exists Z_m, oset_m(Z_m) : (Z_m, oset_m(Z_m)) \in minrules(Z', Z_e, oset(Z')).\end{aligned} \tag{5.29}$$

Define an $allowed\_op$ as any operation that, when applied to any possible $allowed\_state$ of a rule set and rule ordering set, will result in another $allowed\_state$.

Let each element in $minset(Z_m, oset(Z_m))$ consist of all and only those orderings required for a $possibly\_minimal$ rule set Z_m to be minimal, given some prior set of orderings $oset(Z_m)$:

$$\begin{aligned}\forall Z_m \subseteq Z_{combined}, possibly\_minimal(Z_m) = 1,\\ \forall oset(Z_m) \subseteq allset(Z_m) :\\ oset_m(Z_m) \in minset(Z_m, oset(Z_m)) \iff\\ oset(Z_m) \subseteq oset_m(Z_m), minimal(Z_m, oset_m(Z_m)) = 1.\end{aligned} \tag{5.30}$$

It follows directly from the definition of $minrules$ (eq. 5.28) and $minset$ (eq. 5.30) that:

$$\begin{aligned}\forall Z_m \subseteq Z_{combined}, possibly\_minimal(Z_m) = 1, \forall oset(Z_m) \subseteq allset(Z_m) :\\ oset_m(Z_m) \in minset(Z_m, oset(Z_m)) \iff\\ (Z_m, oset_m(Z_m)) \in minrules(Z_m, Z_m, oset_m(Z_m)).\end{aligned} \tag{5.31}$$

Such a $minset(.)$ ordering does not exist for all prior orderings $oset(.)$. The $valid(Z_m, oset(Z_m))$ relation indicates that a specific $oset(Z_m)$ defined with regard to a $possibly\_minimal$ rule set Z_m consists of a subset of the restrictions required by at least one $minset(Z_m, oset(Z_m))$. Specifically:

$$\begin{aligned} \forall Z_m \subseteq Z_{combined}, possibly\_minimal(Z_m) = 1, \\ \forall oset(Z_m) \subseteq allset(Z_m) : valid(Z_m, oset(Z_m)) = 1 \iff \\ \exists oset_m(Z_m) \subseteq allset(Z_m) : oset_m(Z_m) \in minset(Z_m, oset(Z_m)). \end{aligned} \tag{5.32}$$

It follows directly from the definition of $allowed\_state$ (eq. 5.29) and $valid$ (eq. 5.32) that:

$$\begin{aligned} \forall Z_m \subseteq Z_{combined}, possibly\_minimal(Z_m) = 1, \forall oset(Z_m) \subseteq allset(Z_m) : \\ valid(Z_m, oset(Z_m)) = 1 \iff allowed\_state(Z_m, Z_m, oset(Z_m)) = 1. \end{aligned} \tag{5.33}$$

If Z_m is a minimal rule set describing the training data TD, then some of the rules in Z_m will each be a single unique rule, while other rules will each be one of a set of possible options – any one of which could have been generated at a specific point in the rule application order without influencing the number of rules required to predict the training set accurately and completely. Such a combination of rules is referred to as a $rule\_variant$ set.

5.3.2.5 MATCHWORDS, POSSIBLE_WORDS, RULEWORDS AND SHARED_WORDS

Throughout rule extraction, we keep track of the set of words that may influence our decisions with regard to a specific rule. In this way we identify words that match a specific rule ($matchwords$), words that will invoke a specific rule during prediction ($rulewords$), and the set of possible words that may result in rulewords in the final ordering ($possible\_words$). We also identify the possible words that any two rules share ($shared\_words$).

Let the set $matchwords(r)$ consist of all words matched by a specific rule r:

$$\begin{aligned} &\forall r \in Z_{combined}, w \in TD : \\ &w \in matchwords(r) \iff match(w,r) = 1. \end{aligned} \tag{5.34}$$

Let the set $rulewords(Z', oset(Z'), r)$ consist of all words that can cause a specific rule r to be invoked (where the actual rule invoked will depend on the actual rule number assignment), given the current set of rule orderings $oset(Z')$:

$$\begin{aligned} &\forall r \in Z', Z' \subseteq Z_{combined}, w \in TD : \\ &w \in rulewords(Z', oset(Z'), r) \iff \\ &\quad r \in winningrule(Z', oset(Z'), w) \end{aligned} \tag{5.35}$$

Not all rules can necessarily be invoked when predicting the words in TD - for rules that cannot be invoked given the current rule set, the set of $rulewords(.)$ is empty. Note also that the actual words that will invoke rule r in the final ordered rule set consists of the set $rulewords(Z_m, minset(oset(Z_m)), r)$ not the set $rulewords(Z_m, oset(Z_m), r)$.

As the rule set is manipulated, additional rules are added to the required subset Z_e, which can affect the $possible\_words$ sets of all rules later in the rule graph. When comparing two rules r and s, it is possible that rule set extraction has progressed further in a section of the rule graph leading up to one rule than in the section of the rule graph leading up to the other. In order to be able to obtain a clear comparison of the two rules, we choose a shared point in the rule graph (rule v in the definition below) and only allow rules defined prior to this point to influence the $possible\_words(.)$ sets of both rules, resulting in a stable basis for rule comparison.

Let the set $possible\_words(Z', Z_e, oset(Z'), v, r)$ consist of all words that match a specific rule r, and have not yet been caught by rule v or an earlier rule than v, where v in Z_e, the required subset of rule set Z' when rule extraction is in state $Z', Z_e, oset(Z')$:

$$\begin{aligned}\forall r, s \in Z', v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') :\\ w \in possible\_words(Z', Z_e, oset(Z'), v, r) \iff\\ v = r \text{ or } (v, r) \in oset(Z'), match(w, r) = 1,\\ \nexists s \in Z_e : match(w, s) = 1, s = v \text{ or } (s, v) \in oset(Z')^{a}.\end{aligned} \tag{5.36}$$

[a]Note that if $v \neq r$ and $(v, r) \notin oset(Z')$ then $possible\_words(Z', Z_e, oset(Z'), v, r) = \phi$

Let v_0 be an imaginary rule that matches no words, is always the first rule to occur in any rule set, and does not contribute to the rule count of a rule set. The rule v_0 has the following characteristics:

$$v_0 \in Z_{combined}. \tag{5.37}$$

$$|\{v_0\}| = 0. \tag{5.38}$$

$$\forall w \in TD : match(w, v_0) = 0. \tag{5.39}$$

$$\begin{aligned}\forall v_i \in Z', Z' \subseteq Z_{combined}, v_i \neq v_0, \forall oset(Z') \subseteq allset(Z') :\\ (v_0, v_i) \in oset(Z').\end{aligned} \tag{5.40}$$

Since v_0 does not affect further rule set orderings directly, and cannot affect any word-rule relationship, such a rule can be added without causing any side effects in the rule set. We use rule v_0 as a stable point for rule comparison when two rules do not share identical predecessors in Z_e when evaluating $possible\_words$ sets with regard to some stable point v, as defined in eq. 5.36. An alternative stable point that can be used is the last shared parent of the two rules in Z_e, as defined below.

Let $last\_parent(Z', Z_e, oset(Z'), r, s)$ be the latest possible rule or rules in Z_e that occur earlier than both rules r and s according to $oset(Z')$, or the earliest of r and s, if $r, s \in Z_e$:

$$\begin{aligned} &\forall r, s \in Z', v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ &\quad v \in last\_parent(Z', Z_e, oset(Z'), r, s) = 1 \iff \\ &\{(v,r),(v,s)\} \in oset(Z');\ \nexists t \in Z_e : \{(v,t),(t,r),(t,s)\} \in oset(Z'). \end{aligned} \tag{5.41}$$

Let $shared\_words(Z', Z_e, oset(Z'), r, s, v)$ identify those words that are in the $possible\_words$ sets of two different rules r and s with regard to some rule v:

$$\begin{aligned} &\forall r, s \in Z', v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset'(Z') \subseteq allset(Z') : \\ &\quad shared\_words(Z', Z_e, oset(Z'), v, r, s) \equiv \\ &possible\_words(Z', Z_e, oset(Z'), v, r) \cap possible\_words(Z', Z_e, oset(Z'), v, s). \end{aligned} \tag{5.42}$$

5.3.2.6 COMPLEMENTING RULES: CONTAINPAT, MINCOMP AND SUPERCOMP

We now introduce a number of relationships that may exist between pairs of rules. These relationships are crucial in understanding how rules may substitute for one another, and therefore form the foundation for the derivation of minimal rule sets.

Let $complement(Z', Z_e, oset(Z'), v, r, s)$ indicate that rules r and s have overlapping $possible\_words$ sets, i.e.

$$\begin{aligned} &\forall r, s \in Z', r \neq s, v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined} : \\ &complement(Z', Z_e, oset(Z'), v, r, s) = 1 \iff \\ &\quad \exists w \in shared\_words(Z', Z_e, oset(Z'), v, r, s). \end{aligned} \tag{5.43}$$

Let the $path(relation(r, s))$ indicate that a path of relations of a specific type exists between rules r and s in a rule set Z', i.e

$$\forall r, s \in Z' : path(relation(r, s)) = 1 \iff \exists\, x_1 = r, x_2, ..., x_n = s \in Z' :$$
$$relation(x_1, x_2) = relation(x_2, x_3) = ... = relation(x_{n-1}, x_n) = 1. \quad (5.44)$$
$$\forall r, s \in Z' : path(relation(r, s)) = -1 \iff \exists\, x_1 = r, x_2, ..., x_n = s \in Z' :$$
$$relation(x_1, x_2) = relation(x_2, x_3) = ... = relation(x_{n-1}, x_n) = -1. \quad (5.45)$$
$$\forall r, s \in Z' : path(relation(r, s)) = 0 \iff \nexists\, x_1 = r, x_2, ..., x_n = s \in Z' :$$
$$relation(x_1, x_2) = relation(x_2, x_3) = ... = relation(x_{n-1}, x_n) \neq 0. \quad (5.46)$$

where r, s and x_i in the domain of the specific relation.

Let the $path(relation_1/relation_2(r, s)) = -1|0|1$ indicate that a path exists between r and s as defined above, but with edges of either type $relation_1(.)$ or type $relation_2(.)$.

Let the $path(relation_1\&relation_2(r, s)) = -1|0|1$ indicate that a path exists between r and s as defined above, but with edges such that both $relation_1(.)$ and $relation_2(.)$ hold.

Let the relation $containpat(Z', r, s)$ indicate that rule r is a rule with the smallest possible context that contains the context of rule s, i.e.:

$$\forall r, s \in Z', Z' \subseteq Z_{combined} :$$
$$containpat(Z', r, s) = 1 \iff context(r) \supset context(s)$$
$$\text{and } \nexists t \in Z' : context(r) \supset context(t) \supset context(s). \quad (5.47)$$

Let $containpat(Z', r, s) = -1$ if and only if $containpat(Z', s, r) = 1$; and let $containpat(Z', r, s) = 0$ if and only if no $containpat(.)$ relationship exists between r and s in Z'.

From the definition of $containpat$ it follows immediately that

$$\forall r, s \in Z', Z', \subseteq Z_{combined} :$$
$$path(containpat(Z', r, s)) = 1 \iff context(r) \supset context(s). \quad (5.48)$$

Let the bidirectional relation $mincomp(Z', Z_e, oset(Z'), v, r, s)$ be true for all rules r and s that are minimal complements of each other, i.e.

$$\begin{aligned} &\forall r, s \in Z', v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined} : \\ &mincomp(Z', Z_e, oset(Z'), v, r, s) = 1 \iff \\ &complement(Z', Z_e, oset(Z'), v, r, s) = 1, \\ &path(containpat(Z', r, s)) = 0. \end{aligned} \tag{5.49}$$

Let the bidirectional relation $direct(.)$ be true for rules that have either a direct $mincomp(.)$ or a direct $containpat(.)$ relationship, i.e.

$$\begin{aligned} &\forall r, s \in Z', v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined} : \\ &direct(Z', Z_e, oset(Z'), r, s) = 1 \iff \\ &containpat(Z', r, s) = \pm 1 \text{ or } mincomp(Z', Z_e, oset(Z'), r, s) = 1. \end{aligned} \tag{5.50}$$

Let the $subset(Z', Z_e, oset(Z'), v, r, s)$ relation indicate that the $possible\_words$ that can be caught by a rule r is a strict subset of the $possible\_words$ that can be caught by another rule s, with respect to a rule v that occurs earlier in the rule set than either r or s, for a given rule extraction state $Z', Z_e, oset(Z')$:

$$\begin{aligned} &\forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z'), \\ &v \in Z_e, \{(v, r), (v, s)\} \in oset(Z') : \\ &subset(Z', Z_e, oset(Z'), v, r, s) = 1 \iff \\ &shared\_words(Z', Z_e, oset(Z'), v, r, s) \neq \phi, \\ &possible\_words(Z', Z_e, oset(Z'), v, r) \subset possible\_words(Z', Z_e, oset(Z'), v, s). \end{aligned} \tag{5.51}$$

Let the $supercomp(Z', Z_e, oset(Z'), v, r, s)$ relation be true when two rules r and s are both in a $subset(Z', Z_e, oset(Z'), v, r, s)$ and a $mincomp(Z', Z_e, oset(Z'), v, r, s)$ relation:

$$\forall r, s \in Z', v \in Z_e, Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z'),$$

$$supercomp(Z', Z_e, oset(Z'), v, r, s) = 1 \iff \tag{5.52}$$

$$mincomp(Z', Z_e, oset(Z'), v, r, s) = 1,$$

$$subset(Z', Z_e, oset(Z'), v, r, s) = 1. \tag{5.53}$$

For all $r, s \in Z', Z' \subseteq Z_{combined}$, $oset'(Z') \subseteq allset(Z')$: Let $anyset(Z', oset(Z'), r, s)$ be an alternative naming convention for $anyset(Z', Z', oset(Z'), r, s)$, where $anyset$ can be the $subset$ (eq. 5.51), $possible\_words$ (eq. 5.36), or $order\_req$ (eq. 5.58).

5.3.2.7 Z_M AS A SUBSET OF $Z_{COMBINED}$

As mentioned in section 5.3.2.4, the full set of rules in $Z_{combined}$ consists of all possible rules and is therefore a superset of one or more $minimal$ rule sets Z_m. During rule extraction each $allowed$ rule state is defined by a triple $Z', Z_e, oset(Z')$, and each allowed state can can give rise to one or more minimal rule sets $Z_m, oset(Z_m)$, where $(Z_m, oset_m(Z_m)) \in minrules(Z', Z_e, oset(Z'))$.

If any two rules r and s in Z_m have a specific relationship in one such state, this implies further relationships in prior and ensuing states (as shown in statement 15). For any two rules r and s in Z_m it holds that if r and s have a $containpat$ relationship with regard to an appropriate[6] node v when the rule extraction process is in state $Z', Z_e, oset(Z')$, rules r and s will have a similar relationship when rule extraction reaches the state $Z_m, Z_m, oset_m(Z_m)$. It can also be shown that, for any two rules r and s in Z_m, it holds that if r and s have a $complement$ relationship with regard to an appropriate node v when the rule extraction process is in the final state $Z_m, Z_m, oset_m(Z_m)$, then rules r and s will have a similar relationship for each $allowed\_state$ $Z', Z_e, oset(Z')$ leading up to the final state. For $containpat$ path relations, the statement is stronger: Any two rules r and s in Z_m will have a $path(containpat(.))$ relationship when the rule extraction process is in state $Z', Z_e, oset(Z')$ if and only if rules r and s will have a similar relationship when rule extraction reaches the state $Z_m, Z_m, oset_m(Z_m)$.

[6]This v acts as the stable comparison point previously described. Any v such that both (v, r) and (v, s) are included in the set of established orderings would be valid.

5.3.3 RULE ORDERING

In this section, we analyse the conditions and implications of an ordering relationship between two rules in a rule set Z' where $Z' \subseteq Z_{combined}$. We first define what we mean by ordering requirements, and then show how these requirements can be translated to the relationships defined in section 5.3.2.6.

Let $order_{acc}(Z', Z_e, oset(Z_m), r, s)$ indicate that if r does not occur before s, no state $Z', Z_e, oset'(Z')$ can result in an $allowed\_state$, where $oset'(Z')$ is a superset of the $oset(Z')$ orderings:

$$\begin{aligned} &\forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ &\qquad order_{acc}(Z', Z_e, oset(Z'), r, s) = 1 \iff \\ &\nexists oset'(Z') \supseteq oset(Z') \cup (s, r) : allowed\_state(Z', Z_e, oset'(Z')) = 1. \end{aligned} \tag{5.54}$$

Let $order_{red}(Z', oset(Z'), r, s)$ indicate that if rule s occurs before rule r, at least one rule in the rule set Z' will become redundant, irrespective of any further orderings added:

$$\begin{aligned} &\forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ &\qquad order_{red}(Z', Z_e, oset(Z'), r, s) = 1 \iff \\ &\text{at least one rule } t \in Z' \text{ becomes redundant given any state} \\ &(Z', Z'_e, oset'(Z')), oset'(Z') \supseteq oset(Z') \cup (s, r), Z'_e \supseteq Z_e. \end{aligned} \tag{5.55}$$

Let $order(Z', Z_e, oset(Z'), r, s)$ indicate that either an $order_{acc}(Z', Z_e, oset(Z'), r, s)$ or an $order_{red}(Z', Z_e, oset(Z'), r, s)$ relationship holds between any two rules r and s:

$$\begin{aligned} &\forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ &order(Z', Z_e, oset(Z'), r, s) = 1 \iff order_{acc}(Z', Z_e, oset(Z'), r, s) = 1, \\ &\text{or } order_{red}(Z', Z_e, oset(Z'), r, s) = 1. \end{aligned} \tag{5.56}$$

Let $direct\_order(Z', Z_e, oset(Z'), r, s) = 1$ indicate that a direct ordering requirement exists between rules r and s:

$$\begin{aligned} &\forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ &direct\_order(Z', Z_e, oset(Z'), r, s) = 1 \iff order(Z', Z_e, oset(Z'), r, s) = 1; \\ &\nexists t : direct\_order(Z', Z_e, oset(Z'), r, t) = 1, \\ &direct\_order(Z', Z_e, oset(Z'), t, s) = 1. \end{aligned} \tag{5.57}$$

As before, for each of the $order^*$ relations ($order$, $order_{acc}$, $order_{red}$ and $direct\_order$), let $order^*(Z', Z_e, oset(Z'), r, s) = -1$ indicate that $order^*(Z', Z_e, oset(Z'), s, r) = 1$, and let $order^*(Z', Z_e, oset(Z'), r, s) = 0$ indicate that $order^*(Z', Z_e, oset(Z'), r, s) \neq \pm 1$.

Using the above definitions it can be shown that if an $order_{acc}$ relationship exists between any two rules r and s in Z_m, then a path of $complement$ and $direct\_order$ relations exist between these two rules (statement 19). That is, under appropriate conditions, $order_{acc}(Z_m, , Z_m, oset(Z_m), r, s) = 1 \implies path(direct\_order\&complement(Z_m, Z_m, oset(Z_m), v, r, s)) = 1$, where v is a rule earlier than r or s (and in practice typically the $last\_parent$ of these two rules). This means that any two rules in Z_m can only have an accuracy ordering requirement if there exists a path of such complementing direct orderings from one to the other.

Once a $minimal$ rule set Z_m has been obtained, then, if $order_{red}(Z_m, Z_m, oset(Z_m), r, s) = 1$ it also follows that $order_{acc}(Z_m, Z_m, oset(Z_m), r, s) = 1$ (statement 16). The redundancy ordering requirement therefore does not introduce any additional ordering requirements in the final rule graph, but does provide a way to restrict the set of rule orderings early on in the rule extraction process. If any two rules Z' are in a $subset$ relationship with regard to some rule v as above, then it can be shown that these two rules are also in an $order_{red}(Z_m, Z_m, oset(Z_m), r, s)$ relationship with regard to any $Z_m, oset_m(Z_m)$ pair that can be reached from the current $allowed\_state$ (statement 21). Identified $subset$ relationships can therefore be used to define initial orderings prior to further graph manipulation.

The process of applying allowed operations (as discussed in section 5.3.6) leads to further ordering requirements becoming visible. Below we define the conditions under which an $order\_req$ relationship can be enforced between two rules. It can be shown that if two rules r and s are in an $order\_req(Z', Z_e, oset(Z'), v, r, s)$ relationship with regard to a rule v as above, then these two rules are also in an $order_{acc}(Z', Z_e, oset(Z'), r, s)$ relationship (statement 26). The $order\_req$ relationships therefore provide an indication of accuracy ordering requirements that emerge during rule set extraction.

Let the $order\_req(Z', Z_e, oset(Z'), v, r, s)$ relation be true if two rules r and s disagree with regard to outcome, and have a non-empty set of $shared\_words$, all of which agree with rule r with regard to outcome, and at least one of which disagrees with rule s with regard to outcome:

$$\begin{aligned}
\forall r \in Z_{single}, s \in Z', v \in Z_e, Z_e \subseteq Z_m \subseteq Z' \subseteq Z_{combined}, \\
\forall oset'(Z') \subseteq allset(Z') : \\
order\_req(Z', Z_e, oset(Z'), v, r, s) = 1 \iff \\
direct\_order(Z', Z_e, oset(Z'), r, s) = 1, \\
shared\_words(Z', Z_e, oset(Z'), v, r, s) \neq \phi, \\
\forall w_i \in shared\_words(Z', Z_e, oset(Z'), v, r, s) : outcome(w_i) = outcome(r), \\
\exists w' \in shared\_words(Z', Z_e, oset(Z'), v, r, s) : outcome(w') \notin outcome(s).
\end{aligned} \tag{5.58}$$

5.3.4 CHARACTERISTICS OF AN ALLOWED STATE

Let $order\_decided(Z', Z_e, oset(Z'), r, s)$ indicate that the direction of rule ordering between two directly related rules r and s in a rule set Z' has been established based on an identified $containpat$ or $supercomp$ relationship, given some required subset of rules Z_e and some prior set of orderings $oset(Z')$. More specifically:

$$\begin{aligned} \forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset'(Z') \subseteq allset(Z') &: \\ order\_decided(Z', Z_e, oset'(Z'), r, s) = 1 &\iff \\ containpat(Z', r, s) &= 1; \\ \text{or } \exists v \in Z_e : supercomp(Z', Z_e, oset'(Z'), v, r, s) &= 1; \\ \text{or } \exists oset(Z') \subseteq allset(Z') : order\_subset(oset(Z'), oset'(Z')) &= 1, \\ order\_decided(Z', Z_e, oset(Z'), r, s) &= 1. \end{aligned} \tag{5.59}$$

Let $decided\_set(Z', Z_e, oset(Z'))$ take any rule set Z', required subset Z_e and set of rule orderings $oset(Z')$, and generate a set of all the current $order\_decided$ relationships:

$$\begin{aligned} \forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') &: \\ (r, s) \in decided\_set(Z', Z_e, oset(Z')) &\iff \\ path(order\_decided(Z', Z_e, oset(Z'), r, s)) &= 1. \end{aligned} \tag{5.60}$$

Note that an $order\_decided(Z', Z_e, oset(Z'), r, s)$ relationship does not imply that rule r and s will either or both be retained in Z_m, but only that if both were retained, r would be ordered prior to s.

Let $order\_possible1(Z', Z_e, oset(Z'), r, s)$ indicate that, even though it has not yet been established whether a rule ordering is required between two rules r and s: if a rule ordering is required, the direction of such an ordering will be from rule r to rule s because of the existence of a $order\_req$ relationship between these two rules. More specifically:

$$\begin{aligned} \forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ order\_possible1(Z', Z_e, oset(Z'), r, s) = 1 \iff \\ \exists v \in Z_e, \{(v, r), (v, s)\} \in oset(Z') : order\_req(Z', Z_e, oset'(Z'), v, r, s) = 1. \end{aligned} \tag{5.61}$$

Let $order\_possible(Z', Z_e, oset(Z'), r, s)$ indicate that the direction of rule ordering (if any) has not yet been established between two minimal complements r and s that have remaining $shared\_words$ when rule extraction is in state $Z', Z_e, oset(Z')$. More specifically:

$$\begin{aligned} \forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, \forall oset(Z') \subseteq allset(Z') : \\ order\_possible(Z', Z_e, oset(Z'), r, s) = 1 \iff mincomp(Z', r, s) = 1, \\ shared\_words(Z', Z_e, oset(Z'), r, s) \neq \phi, \\ order\_decided(Z', Z_e, oset(Z'), r, s) = 0. \\ order\_possible1(Z', Z_e, oset(Z'), r, s) \neq -1. \end{aligned} \tag{5.62}$$

Let $possible\_set(Z', Z_e, oset(Z'))$ take any rule set Z', required subset Z_e and set of rule orderings $oset(Z')$, and generate a set of all the $decided$ and $possible$ rule orderings. More specifically:

$$\begin{aligned} \forall r, s \in Z', Z_e \subseteq Z' \subseteq Z_{combined}, oset(Z') \subseteq allset(Z') : \\ (r, s) \in possible\_set(Z', Z_e, oset(Z')) \iff \\ path(order\_possible/order\_decided(Z', Z_e, oset(Z'), r, s)) = 1. \end{aligned} \tag{5.63}$$

Consider any $Z', Z_e, oset(Z')$ combination such that $allowed\_state(Z', Z_e, oset(Z')) = 1$. By definition (eq. 5.28 and eq. 5.29) it will always hold that:

$$Z_e \subseteq Z_m \subseteq Z_{combined}. \tag{5.64}$$

Furthermore, it can be shown (statement 28 and 27) that:

$$order\_subset(decided\_set(Z', Z_e, oset(Z')), oset_m(Z_m))) = 1. \tag{5.65}$$

$$order\_subset(oset_m(Z_m), possible\_set(Z', Z_e, oset(Z'))) = 1. \tag{5.66}$$

Now consider any two rules r and s that will be retained in the minimal rule set Z_m. From the above it follows that if rules r and s are ordered according to $decided\_set(Z', Z_e, oset(Z'))$ then these two rules will retain this ordering in the minimal rule set ordering $oset_m(Z_m)$. Also, any rule ordering that will eventually be required in $oset_m(Z_m)$ is currently listed in $possible\_set(Z', Z_e, oset(Z')$. For any given state $Z', Z_e, oset(Z')$, the definite and possible rule orderings can therefore be generated automatically, and used to reason about further graph manipulation options.

5.3.5 INITIAL ALLOWED STATE

The rules in $Z_{combined}$ describe the training data completely, but not necessarily accurately. Since all possible rules are included in $Z_{combined}$, it follows that $\phi \subseteq Z_m \subseteq Z_{combined}$ for all $minimal$ rule sets Z_m. Furthermore, it can been shown that if the rules in $Z_{combined}$ are ordered according to $containpat$ and $supercomp$ relations, then no overly restrictive orderings will be added. If the rule set $Z_{combined}$ is ordered according to the rule set orderings generated by $decided\_set(Z_{combined}, \phi, \phi)$, then the rule set is in an $allowed\_state$ (statement 33). This state is used as the initial state prior to application of the various $allowed\_ops$, as described below.

5.3.6 ALLOWED OPERATIONS

Each $allowed\_op$ as defined in section 5.3.2.4 changes the state of the rule set, required rule subset and rule ordering set, from one $allowed\_state$ to another, with the initial $allowed\_state$ defined in section 5.3.5. These operations are not unique, and both stronger and weaker versions can be constructed. While the framework up to this point has been defined rigorously, we now discuss a number of possible operations in order to demonstrate how this framework can be used during rule extraction. We describe a number of operations that we have implemented and tested in our rule extraction system. Specifically, we describe allowed operations that can be applied to (1) delete rules, (2) remove unnecessary edges, (3) mark rules as required, and (4) resolve conflicted rules.

When applying any of these operations it is assumed that the rule graph is in an $allowed\_state$ defined by the triple $Z', Z_e, oset(Z')$. Prior to discussing these operations in further detail, it should be noted that the rule graph edges added according to the $decided\_set$ and $possible\_set$ orderings have two functions: On the one hand, these edges define the order in which rules will occur, as discussed up to this point. Secondly, these edges link any rule to *all possible rules* that may potentially replace or be replaced by the current rule. *Direct successors* or *predecessors* are identified by following both $decided\_set$ and $possible\_set$ orderings and identifying rules that either have to or can occur directly before or after the rule being considered. From these sets of rules it is possible to define the total number of rules that (1) will definitely and (2) may potentially (based on decisions made in other sections of the rule graph) be deleted if a current rule is associated with a specific outcome.

5.3.6.1 DECREASING RULE SET SIZE

A rule r can only be deleted if it can be shown that the rule has become redundant, and will remain redundant. This can occur for two reasons: (1) because of the position of the specific rule r in the rule graph, all words that match this rule are already caught by required rules (correctly identified as such) that occur earlier in the rule set; or (2) because the rule can be 'merged' with a second rule occurring at the same point in the rule extraction order. We define three different allowed operations with regard to rule deletion:

1. A rule r may be deleted if some rule s exists such that:
   - Rules r and s are resolved ($r, s \in Z_{single}$) and agree with regard to outcome.
   - Rules r and s have identical relationships with identical predecessors (both possible and definite).
   - Rules r and s have identical relationships with identical successors (both possible and definite).
2. A rule r may be deleted if a set of rules v_i exists such that:
   - The v_i constitute all the direct successors of rule r.
   - Rule r and all the v_i are resolved ($r, v_i \in Z_{single}$) and agree with regard to outcome. (No rule t with a potentially conflicting outcome can have an ordering that allows it to occur between rule r and any v_i.)
3. A rule r may be deleted if for some allowed v, the $possible\_words(Z', Z_e, oset(Z'), v, r) = \phi$.

5.3.6.2 REMOVING UNNECESSARY EDGES

Since $decided$ orderings are transitive, it is possible to remove any explicit ordering $rule\_order(Z', r, s)$ if it already holds that $rule\_order(Z', r, t) = 1$ and $rule\_order(Z', t, s) = 1$ for some $t \in Z'$. Note that this does not remove (r, s) from $oset(Z')$.

5.3.6.3 IDENTIFYING REQUIRED RULES

When a rule is identified as *required*, it is moved from the possible rule set Z' to the set of required rules Z_e. This is an $allowed\_op$ for a rule r if the rule r itself has already been resolved (that is, it can only predict a single outcome), and no further rules exist that may potentially agree with regard to outcome.

5.3.6.4 RESOLVING CONFLICT RULES

We define four different operations that can be used to resolve conflicted rules. All of these operations utilise the concept of a $single$ rule and a $conflict\_count$. A rule s is identified as a $single$ rule if there exists at least one word w such that $(w, s) \in oset(Z')$ and no t exists such that $match(w, t) = 1$, unless $(s, t) \in oset(Z')$. This means that word w can only be predicted by rule s, or by a later rule that has a decided path from word w via rule s. For each possible outcome we count the number of predecessors that will definitely be deleted if rule r is resolved to that specific outcome ($definite\_count$), as well as the number of predecessors that may possibly be deleted ($possible\_count$). These counts are not calculated per predecessor, but rather per word set that a predecessor represents. For a predecessor s to contribute to a ($definite\_count$), the predecessor must be resolved (that is, $s \in Z_{single}$), be identified as a $single$ rule, and have only one successor (the conflicted rule r itself).

1. If it is clear that rule r will provide an advantage if resolved to a specific outcome, resolution is performed, and the conflicted rule is replaced with a normal rule with the selected outcome. A rule may only be resolved in this way, if the $definite\_count$ for a specific outcome dominates the sum of the $definite\_count$ and $possible\_count$ for all alternative outcomes. It is also required that at least one of the predecessors with a outcome matching the outcome selected for rule resolution be a $single$ rule. This prevents an unnecessary rule from being generated at this point in the rule application order. (In a later step, the resolved conflict node will merge with the $single$ predecessor.)

2. If a conflicted rule r has no predecessors that can potentially agree with each other with regard to a specific outcome, the rule r may be deleted. (We refer to this process as a *lost conflict*). A rule may be resolved in this way if the sum of the $definite\_count$ and $possible\_count$ is less than or equal to one, for all possible outcomes.

3. If for any of the outcomes the sum of the $definite\_count$ and $possible\_count$ is less than or equal to one, that outcome may be removed from the possible outcomes of the conflicted rule, even though the rule remains a conflicted rule. If all except one outcome are removed in this way, then at least one predecessor must be a $single$ rule that agrees with regard to the final outcome, as discussed with regard to the previous operator.

4. The root node is not allowed to be resolved via operator (2) or (3). If operator (1) is not applicable, the root node is resolved to the outcome that occurs most in the training data. This operation is only allowed when all predecessors have been resolved.

5.3.7 BREAKING TIES

Once all the possible operations have been applied to the rule graph, and no further simplification is possible, it does not necessarily mean that all conflicted rules have been resolved. The extent to which the rule graph is resolved, depends on the strength of the various allowed operations defined. Note that the set difference between the increasing rule set Z_e, and the decreasing rule set Z' provides a clear indication of the extent to which the current solution still falls short of the minimal rule set Z_m. If Z_e equals Z', a minimal rule set has been obtained.

The remaining conflicted nodes can be solved by viewing the rule graph as a constraint satisfaction problem (CSP). By assigning the various remaining (node-specific) outcome values to each of the remaining conflicted nodes, and searching through the resulting search space, the final solution can be obtained. During the CSP search process, all conflicted nodes are solved simultaneously, and the rule minimisation process proceeds rapidly using the various deletion operations. By searching through all the remaining allowed rule sets, the smallest possible set can be obtained.

The magnitude of this CSP is determined by the coverage of the operations employed. If only trivial operations were employed, and all conflicts were left to the CSP to resolve, a huge CSP would result for even very small problems. The stronger the allowed operations defined, the smaller the CSP to solve. Our current implementation has been used to solve small tasks of $n = 100$ words, and we have been able to extract rule sets that are smaller than that extracted by *Default&Refine*. Various CSP-specific techniques can be applied to improve the computational tractability of the task. However, this is not the focus of the current chapter, which aims to define a solid theoretical basis for further experimentation: computational tractability will be addressed in future work.

5.3.8 OPTIMISING GENERALISATION ABILITY

Once all conflicted rules have been solved, and the minimal rule set obtained, it is possible to refine the rule set by choosing the best rule option among the various variants available. Possible selection criteria include smallest rule context, most symmetric rule context, best coverage of the training data, best fallback given the following set of rules in the specificity hierarchy, and various others. Since the choice of variant does not influence the number of rules generated, this provides flexibility in the construction of the final rule set. When heuristics are employed during rule extraction, choices are limited earlier on: this framework allows heuristic choices to be postponed as late as possible during rule extraction, and makes those choices explicit.

5.4 ALTERNATIVE ALGORITHMS AS SPECIALISATION OF GENERAL FRAMEWORK

It is interesting to note that the rule extraction algorithms discussed in Chapter 4 can be viewed within the *minimal representation graph* framework. In the case of DEC the full set of rules in Z is not constructed: only the rules that match the DEC format are generated. The rule graph is not constructed prior to rule extraction but is grown during rule extraction according to $containpat$ relationships. Each additional conflicting word results in another leaf being added to the graph.

Default&Refine also grows the graph from the root outwards, ordering rules according to rule extraction order. At each level, a decision is made with regard to the rule and associated outcome that best predicts the set of words that must be caught at that level in the rule graph (that will not be predicted correctly by a later rule). This is conceptually similar to generating a full rule graph prior to rule extraction, and resolving rules strictly from the root outwards according to a greedy heuristic at each level. Since neither algorithm proceeds with allowed operations from an allowed initial state, both will in general produce larger-than-minimal rule sets.

5.5 EXTENSIONS

The current framework provides a solid theoretical base for reasoning about the choices made during grapheme-to-phoneme rule extraction. We are interested in how this framework can be extended to incorporate additional techniques, and this will be addressed in future work. Specific extensions that may fit well within this framework include:

- Pronunciation variants: currently pronunciation variants are not allowed (See eq. 5.13). If a single pseudo-phoneme is generated for each alternating sound, the same framework can be used to process pronunciation variants, with variants expected to drift towards the top of the graph, unless clearly systematic. Further choices ensue with regard to resolving conflict between a single phoneme and a matching pseudo-phoneme, and extensions to the current framework may assist in resolving such issues.

- Class-based groupings: It is clear from rule set analysis that groups of graphemes tend to influence neighbouring graphemes in systematic ways. It should be possible to accelerate the learning process by extracting such graphemic groups during rule set extraction. This may require the interlinking of a number of minimal memory graphs in a single structure.

- Combining phonemic and graphemic context: The same rule set can be generated in terms of either graphemic or phonemic context. We are interested in the advantages and disadvantages of combining both approaches in a single rule set.

- Graphemic chunks: As all possible word sub-components are generated during rule set extraction, the extent to which the rule graph is manipulated brings this approach closer or removes it further from pronunciation-by-analogy techniques. We are interested in the similarities and differences between these two approaches.

5.6 CONCLUSION

In this chapter we described a theoretical framework that can be used to analyse the grapheme-to-phoneme prediction problem in a rigorous fashion. Using this framework, it is possible to define a number of ‘allowed operations’ that attempt to extract the smallest possible rule set from any given set of training data. By making the various options available at each stage of rule extraction explicit, we obtain a better understanding of the grapheme-to-phoneme prediction task itself. Furthermore, the new framework provides a solid foundation for further research in pronunciation prediction, including the potential incorporation of pronunciation variants, class-based groupings and/or graphemic chunks within a rewrite-rule based framework.