Helsinki University of Technology Laboratory for Theoretical Computer Science
Technical Reports 20
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tekninen raportti 20
Espoo 2001 HUT-TCS-B20

A TREE EXPANSION FORMALISM FOR GENERATIVE STRING REWRITING

Eero Lassila

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory for Theoretical Computer Science
Technical Reports 20
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tekninen raportti 20
Espoo 2001 HUT-TCS-B20

A TREE EXPANSION FORMALISM FOR GENERATIVE STRING REWRITING

Eero Lassila

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietojenkäsittelyteorian laboratorio

Distribution:
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O.Box 5400, FIN-02015 HUT, Finland
Tel. +358-0-4511
Fax. +358-0-451 3369
E-mail: lab@tcs.hut.fi

© Eero Lassila
June 2001

ISBN 951-22-5554-5
ISSN 0783-540X

Printing:
Picaset Oy
Helsinki 2001

ABSTRACT: An abstract generative string rewriting device is introduced for modeling such practical program generation and optimization tools as macro processors. The novel device is called a monosystem, and it differs from such classical abstract generative devices as Chomsky grammars (which have more applications in program analysis than in program generation) by being able to operate on an infinite alphabet and by being sensitive to an unbounded two-sided context. The infinite alphabet makes it conceptually simple to deal with structured symbols (such as macro calls), and the unbounded context-sensitivity promotes optimization. An extension (called a trisystem) of the basic monosystem model is capable of emulating both type-1 Chomsky grammars and a variety of Lindenmayer systems.

A monosystem divides into a rule base and a separate control mechanism. The whole rule base is represented as a single function called a letter-refiner, which is capable of replacing any symbol occurrence in any context with an appropriate new substring. Since only one symbol occurrence is thus rewritten at a time, it is natural to record the rewriting history in a tree form. The rewriting context for each tree leaf is extracted from some cross section of the same tree. The actual cross section is determined by a function called a belt-selector, which is the single user-adjustable parameter of the control mechanism.

The central problem with monosystems is to guarantee that the output has the intended semantics. If nothing else is known about the letter-refiner than that it is semantics-preserving, the belt-selector must not allow parallelism (and more specifically, the rewriting context of any tree leaf must equal the tree frontier, that is, the set of all the contemporary leaves). When the letter-refiner fulfills a sufficiently strong additional constraint, synchronous and even asynchronous parallelism become possible. Moreover, it then becomes possible to switch between sequential and parallel rewriting (and even between various subtypes of either one), without any need to modify the letter-refiner. These switches may change the structure but not the semantics of the output.

This preliminary report is best seen as a tutorial. However, a self-contained formalization of the monosystem model and its properties is included as a set of appendices. As yet, the main emphasis is on definitions rather than on theorems, and all proofs are accordingly omitted.

CONTENTS

1 Introduction **1**
- 1.1 Structure of this report 5

2 Trees, tree belts, and belt-selectors **6**
- 2.1 Trees 6
- 2.2 Angles between tree nodes 7
- 2.3 Belts and belt-selectors 8
- 2.4 Combs 11
- 2.5 Four important belt-selectors: σ_{E}, σ_{C}, σ_{I}, and $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ 12
- 2.6 On the chosen tree representation 12
- 2.7 Pointers into the appendices 15

3 Monosystems **16**
- 3.1 Letters and words 16
- 3.2 Letter-refiners 16
- 3.3 Constituents of a monosystem 17
- 3.4 Belt-selector stagnancy 18
- 3.5 Operation of a monosystem 18
- 3.6 Pointers into the appendices 20

4 Emulating other devices by monosystems **22**
- 4.1 Macro processors and $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ 22
- 4.2 Parametric Lindenmayer systems and σ_{C} 22
- 4.3 ReFlEx prototype and σ_{I} 23

5 Properties of monosystems **24**
- 5.1 Confluence 24
- 5.2 Progressiveness 25
  - 5.2.1 Weak progressiveness 25
  - 5.2.2 Strong progressiveness 26
  - 5.2.3 Distributive progressiveness 27
- 5.3 Stableness 27
- 5.4 Soundness 29
  - 5.4.1 Assuming only letter-refiner soundness 29
  - 5.4.2 Imposing legitimacy constraints on the letter-refiner . 30
- 5.5 Coalescence 31
- 5.6 Pointers into the appendices 32

6 On trisystems **34**
- 6.1 Definition of a trisystem 34
- 6.2 Properties of trisystems 35
- 6.3 Emulating other devices by nondeterministic trisystems 36
  - 6.3.1 Chomsky grammars and frame $\langle\sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}, \sigma_{\mathsf{E}}\rangle$ 36
  - 6.3.2 Lindenmayer systems and frame $\langle\sigma_{\mathsf{C}}, \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}}\rangle$ 37

7 Conclusion **38**

References 40

A Mathematical preliminaries 42
A.1 General on notation and terminology 42
A.1.1 Sets 42
A.1.2 Cartesian products, binary relations, and functions . . 42
A.1.3 Finite sequences 43
A.2 Properties of binary relations 43
A.3 Letters and words 44

B Expansion trees 45
B.1 Structure of an expansion tree 45
B.2 Basic relations between tree nodes 45
B.3 More relations between tree nodes 46
B.4 Node altitudes and node pair angles 47
B.5 Tree orthoextensions 48

C Tree belts and belt-selectors 49
C.1 Belts 49
C.2 Belt projection 49
C.3 Belt wrapping 49
C.4 Belt-selectors 50
C.5 Wings of a belt-selector 50

D Monosystems 51
D.1 Letter-refiners 51
D.2 Constituents of a monosystem 51
D.3 Belt-selector stagnancy 52
D.4 Tree expansion by leaf unfolding 52
D.5 Output extraction 53
D.6 On confluence 53
D.7 On the infertileness of the immutables 54

E More on belt-selectors 55
E.1 Links 55
E.2 Characteristic comb of a belt-selector 55
E.3 Locks and hooks of a belt-selector 56
E.4 Operations on belt-selectors 57
E.4.1 Junction of two belt-selectors 57
E.4.2 Mirror-image 57
E.4.3 Incrementation 57
E.4.4 Convexification 57
E.5 Belt-selector wrapping 58
E.6 Some individual belt-selectors 59

F Progressiveness 60
F.1 Weak progressiveness 60
F.2 Strong progressiveness 61
F.3 Distributive progressiveness 61
F.4 Rigidness 62

F.5 Further results 62
F.6 Idealness 63

G Stableness **64**

H Soundness **65**
H.1 Basic definitions and results 65
H.2 Rewriting maps 65
H.3 Reflexive, transitive, and adjunctive maps 66
H.4 Civil, semicivil, and semigentle maps 66
H.5 On smooth belt-selectors 67
H.6 On subideal belt-selectors 67

I Coalescence **69**

1 INTRODUCTION

Computerized information processing often involves manipulation of finite strings of symbols. For example, computer programs themselves, when interpreted as data, are finite instruction sequences, and their compile-time generation and optimization may be seen as string manipulation. In addition to atomic symbols like the characters in a character string, structured symbols are accordingly allowed to occur in the strings considered: we definitively want to be able to choose so high a level of abstraction that computer programs are treated as instruction sequences rather than as character sequences.

We are especially interested in the case in which the elementary string manipulation operations available are what we call *refinements*: one symbol occurrence is replaced with an appropriate new substring, as depicted in Figure 1.

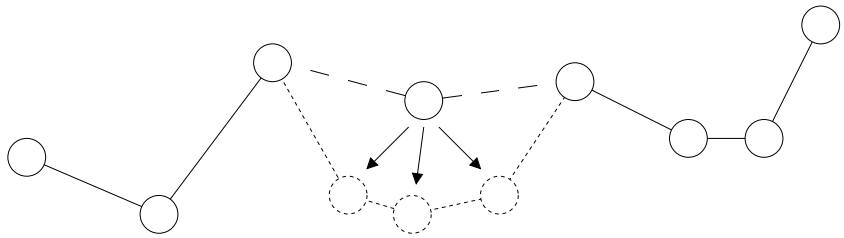

Figure 1: A refinement.

By introducing an auxiliary root node, we are able to represent an arbitrary series of successive refinements as a *history tree*, as suggested in Figure 2. The particular tree in the figure is seen to be the result of five successive *leaf unfoldings*, each of which records a single refinement. Obviously, the tree representation partially hides the actual order in which the refinements are performed.

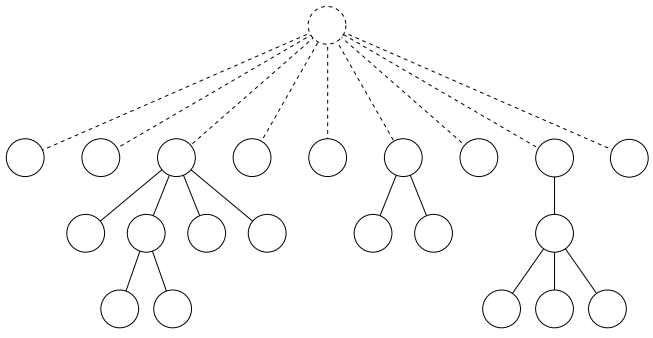

Figure 2: A series of refinements represented as a history tree.

For the purpose of tasks like optimizing code generation, the refinements should be context-sensitive, and furthermore, the effective context should be unbounded in both directions. The string to be used as the *refinement context* is extracted from such a cross section of the history tree that contains the particular leaf to be unfolded; the selected cross section is called the

unfolding context. (So our "unfolding context" is a sequence of tree nodes, whereas our "refinement context" is a sequence of symbols. However, there is little need to be afraid of mixing up these two terms, since whenever one of them is used below, its denotation is usually obvious.) Even if it might perhaps seem natural to require that the unfolding context always equals the *frontier*, that is, the set of all the contemporary leaves of the history tree, we specifically do not adopt this restriction.

EXAMPLE 1. ▽

Consider the tree in Figure 3, and suppose that the next leaf to be unfolded is the only one with symbol `M`, marked with a black ring. Assuming that the unfolding context of this leaf is constituted by the nodes marked with white rings, we notice that the two sides of the refinement context of this instance of `M` are `PWT` and `UN`.

The designated unfolding context obviously differs from the frontier. If the unfolding context equaled the frontier, then the two sides of the refinement context would be `PWR` and `ZQQ`. △

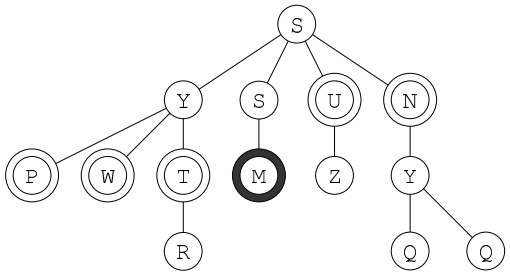

Figure 3: An unfolding context that differs from the frontier.

Next we look at three program generation tools. The unfolding context used by the first tool always equals the frontier, which is not the case with the other two tools.

- Consider *macro processors* [2, 4, 3, 5]. The expansion of each macro call (which we interpret as a single structured symbol), that is, each refinement, may be sensitive to the current values of any *global macro-time variables* [3, pp. 61–64], and these values customarily propagate from left to right. This implies that at each tree leaf unfolding, all the leaves on the left-hand side must possess terminal symbols rather than other macro calls; otherwise, the values of the variables could not be properly evaluated. Therefore, the leaf processing order is strictly depth-first and left-to-right, and the unfolding context necessarily equals the frontier (despite the fact that macro processors actually ignore the right-hand side of the refinement context).

  Figure 4 depicts a representative example of the unfolding context selection scheme: the cross section selected for the node with the black ring is again indicated by white rings. (Notice that this particular figure includes all the nodes of the final history tree, even if they are not yet present when the designated node is unfolded.)

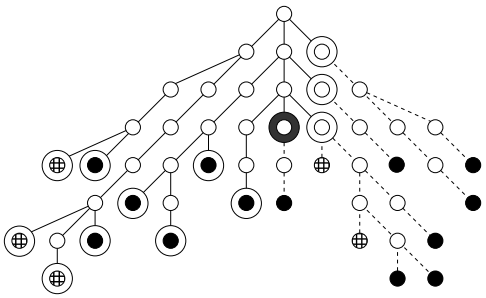

Figure 4: The unfolding context used by macro processors.

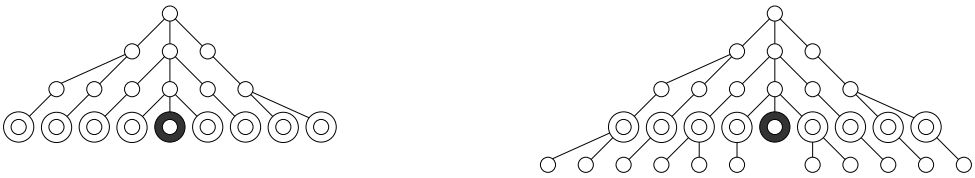

Figure 5: The unfolding context used by Lindenmayer systems.

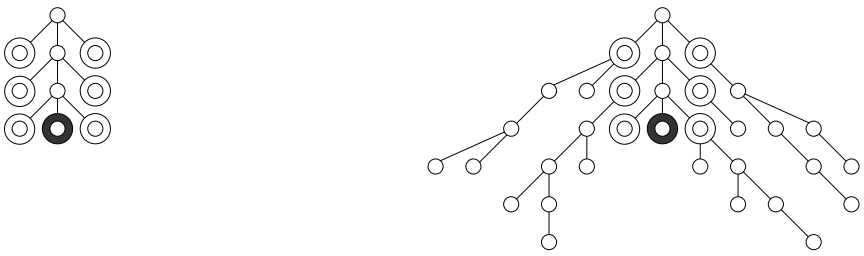

Figure 6: The unfolding context used by the ReFlEx prototype.

- *Parametric Lindenmayer systems* [24, 23, 18, 17, 22, 21] output sequences of picture drawing commands and thus indirectly produce high-quality graphics. They are perhaps the best-known example of application-oriented extensions to the classical *Lindenmayer system* model [8, 25, 9] of formal language theory: like macro processors but unlike classical Lindenmayer systems, parametric Lindenmayer systems support the use of structured symbols.

  With Lindenmayer systems (whether parametric or not), the tree nodes are unfolded in a generation-by-generation fashion, and the "horizontal" cross section constituted by all the nodes in the current generation serves as the unfolding context. In practice, the nodes within each single generation may well be unfolded sequentially in any order, rather than simultaneously, but then it should be taken care that the unfolding context is still given by the horizontal cross section and not by the frontier.

  The two trees of Figure 5 depict only the two extremes among the possible unfolding moments of the leaf with the black ring.
- Figure 6 illustrates the unfolding context used by an utterly simplistic prototype, called *ReFlEx* [10, 11, 12], of a still nonexistent tool proposed by us for optimizing machine-level code generation [1, 14, 20, 19]. With this unfolding context, the leaf processing order becomes completely free. The two trees of Figure 6 again depict only two of the possible unfolding moments of the leaf with the black ring. (The moment depicted on the left is of course the earliest possible.)

Our present goal is to devise a general abstract model to capture such concrete string generation tools as macro processors, parametric Lindenmayer systems, and the ReFlEx prototype. The model is to have two main components: a *rule base* specifying the acceptable refinements (which are typically unboundedly context-sensitive); and a separate *control mechanism* driving the application of the rules. Unlike the rule base, the control mechanism should, in general, be aware of the history tree and, in particular, determine the tree cross section to be used as the unfolding context. The following two aspects of the unfolding context selection appear especially interesting.

- *Progressiveness.*
  Macro processing is strictly sequential whereas the Lindenmayer rewriting is synchronously parallel, and we claim that the ReFlEx rewriting is asynchronously parallel. Which unfolding context selection schemes enforce a sequential leaf processing order, and which ones allow synchronous or even asynchronous parallelism? On the other hand, can we find out whether a given scheme may result in premature halting, that is, halting before all the history tree leaves possess terminal symbols?
- *Soundness.*
  When the final string consisting of terminal symbols is interpreted as, say, a program, can we be sure that it has the intended semantics? If the unfolding context always equals the frontier (as with macro processing), it seems sufficient that each single refinement rule is

semantics-preserving: after each leaf unfolding, the string contained by the new frontier still has the intended semantics. But what if the unfolding context differs from the frontier (as with the Lindenmayer rewriting or with the ReFlEx rewriting)?

1.1 Structure of this report

In the remaining Sections 2–7, we give a detailed but still somewhat informal description of the abstract model suggested above. Section 2 first focuses on the key component of the control mechanism of the model, and Section 3 then presents the structure and operation of the full model. The expressive power of the model is examined in Section 4, where we return to the three concrete string generation tools discussed above. Section 5 scrutinizes some interesting internal properties (such as progressiveness and soundness) of the model, and Section 6 sketches a possible extension to the model. Finally, some concluding remarks are presented in Section 7.

A completely formal definition of the model is included but only as a set of appendices, that is, Sections A–I. More specifically, the appendices provide a backbone for Sections 2, 3, and 5; and each one of these three earlier sections accordingly ends with a subsection called 'Pointers into the appendices'. The presentation in the appendices is self-contained but rather concise.

The emphasis of this report is on definitions rather than on theorems. As yet, all our actual claims are preliminary conjectures only: even if they are explicitly marked as 'Theorems' or 'Propositions', their proofs are omitted.

2 TREES, TREE BELTS, AND BELT-SELECTORS

The control mechanism of our string generation tool model will be described in its entirety in Section 3. Here we take a preliminary step by defining *belt-selectors*: the control mechanism has a single user-adjustable parameter, which is a belt-selector. Belt-selectors constitute a particular subclass of such functions that take a given node in a given tree and return such a cross-section-type subset, or a *belt*, of the tree nodes that contains the argument node. (Belt-selectors were originally introduced in [13], where a proof of Theorem 8 below can be found.)

2.1 Trees

A *tree* consists of a finite and non-zero number of *nodes*. Each tree is *rooted* and *ordered*, as will soon be explained. (This denotation of the term 'tree' adopted here is a standard one; see [6], for instance.) Example 2A introduces a sample tree, which will be utilized by several examples in the present Section 2.

EXAMPLE 2A. ▽

Let the tree in Figure 7 be called A. By convention, "node a_3", for instance, refers to the unique node in tree A labeled as '3'. The reason why node a_9 is distinguished is that we have, more or less arbitrarily, chosen it to have an important role in some examples below. △

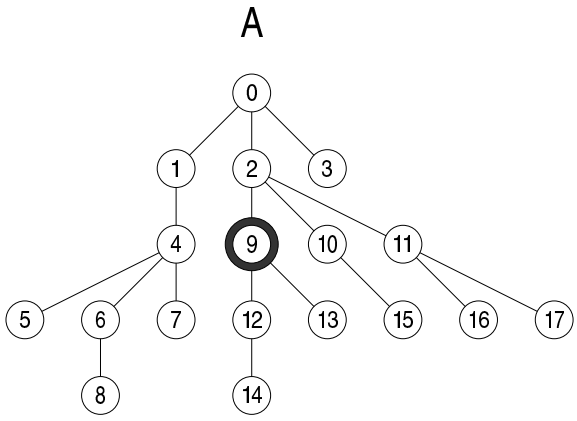

Figure 7: Tree A.

The rootedness means that each tree has exactly one *root*. Every tree node different from the root has exactly one *father* in the tree and is called a *son* of the father. Such tree nodes that have no sons are *leaves* of the tree, and the *frontier* of the tree is the set consisting of all the leaves.

We say that a given node n' is an *ancestor* of a given node n if the pair $\langle n', n\rangle$ belongs to the reflexive-transitive closure of the binary 'is a father of' relation. (By "a pair" we always mean an ordered pair.) If n' is an ancestor of n, then n is a *descendant* of n'. Moreover, n' is a *proper ancestor* of n, and n is correspondingly a *proper descendant* of n', if n' is an ancestor of n and

$n' \neq n$.

EXAMPLE 2B. ▽

The root of tree A is node a_0, and the frontier of A consists of all the nine leaves a_5, a_8, a_7, a_{14}, a_{13}, a_{15}, a_{16}, a_{17}, and a_3.

The father of a_9 is a_2, and so a_9 (like a_{10} and a_{11}) is a son of a_2. The ancestors of a_9 are a_9, a_2, and a_0 (of which the last two are proper ancestors). △

The orderedness means that there is a total "left-to-right" order among the sons of any given tree node. If two distinct nodes have the same father, then one of them is a *left-brother* of the other, and the latter is a *right-brother* of the former.

We say that a given node n' is a *left-relative* of a given node n if there are such nodes n'_0 and n_0 in the tree that n'_0 is a left-brother of n_0, n' is a descendant of n'_0, and n is a descendant of n_0. If n' is a left-relative of n, then n is a *right-relative* of n'.

EXAMPLE 2C. ▽

In tree A, node a_9 has right-brothers a_{10} and a_{11}, and a_{10} has a_9 as a left-brother and a_{11} as a right-brother.

Node a_8 is an example of a left-relative of a_9: a_1, which is an ancestor of a_8, is a left-brother of a_2, which is an ancestor of a_9. Correspondingly, a_9 is a right-relative of a_8. △

Note that for each two distinct nodes n_1 and n_2 in any given tree, exactly one of the following statements holds: n_1 is a proper ancestor of n_2; n_1 is a proper descendant of n_2; n_1 is a left-relative of n_2; or n_1 is a right-relative of n_2.

2.2 Angles between tree nodes

Each tree node has a unique *altitude*, and each pair of tree nodes has a unique *angle*.

Definition 1. The ***altitude*** of a given tree node is the number of its proper ancestors.

Definition 2. The ***angle*** of a given tree node pair $\langle n, n' \rangle$ is denoted as $\sphericalangle(n, n')$ and defined as the unique integer triple $\langle i, d, j \rangle$ that meets the following conditions.

1. i [respectively, j] is the difference of the altitudes of n [respectively, n'] and the one of the common ancestors of n and n' that has the greatest altitude.
2. $d = 0$ if one of n and n' is an ancestor of the other, $d = -1$ if n' is a left-relative of n, and $d = 1$ if n' is a right-relative of n.

Note that $\sphericalangle(n, n') = \langle i, d, j \rangle$ always implies $\sphericalangle(n', n) = \langle j, -d, i \rangle$. Note also that a given triple $\langle i, d, j \rangle$ may appear as a node pair angle if and only if all the following conditions are met: $i \geq 0$, $d \in \{-1, 0, 1\}$, $j \geq 0$, and

$d = 0 \Leftrightarrow i \cdot j = 0.$

EXAMPLE 2D. ▽

Table 1 lists the angles from node a_9 to the other nodes of tree A. △

n	$\sphericalangle(\mathsf{a}_9, n)$	n	$\sphericalangle(\mathsf{a}_9, n)$	n	$\sphericalangle(\mathsf{a}_9, n)$
a_0	$\langle 2, 0, 0\rangle$	a_6	$\langle 2, -1, 3\rangle$	a_{12}	$\langle 0, 0, 1\rangle$
a_1	$\langle 2, -1, 1\rangle$	a_7	$\langle 2, -1, 3\rangle$	a_{13}	$\langle 0, 0, 1\rangle$
a_2	$\langle 1, 0, 0\rangle$	a_8	$\langle 2, -1, 4\rangle$	a_{14}	$\langle 0, 0, 2\rangle$
a_3	$\langle 2, 1, 1\rangle$	a_9	$\langle 0, 0, 0\rangle$	a_{15}	$\langle 1, 1, 2\rangle$
a_4	$\langle 2, -1, 2\rangle$	a_{10}	$\langle 1, 1, 1\rangle$	a_{16}	$\langle 1, 1, 2\rangle$
a_5	$\langle 2, -1, 3\rangle$	a_{11}	$\langle 1, 1, 1\rangle$	a_{17}	$\langle 1, 1, 2\rangle$

Table 1: The angles from node a_9 to the other nodes of tree A.

2.3 Belts and belt-selectors

Definition 3. A ***belt*** of a tree is any such subset of the tree nodes that each leaf of the tree has exactly one ancestor in the subset.

In other words, a tree belt is a cross section of the tree. In any tree, both the frontier and the set consisting of the sole root are belts. For some less trivial belt instances, see Example 2E.

EXAMPLE 2E. ▽

Table 2 lists all such belts of tree A that contain node a_9. △

$\{\mathsf{a}_1\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{11}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_6, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{11}, \mathsf{a}_3\}$
$\{\mathsf{a}_1\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{11}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_6, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{11}, \mathsf{a}_3\}$
$\{\mathsf{a}_1\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_6, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$
$\{\mathsf{a}_1\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_6, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$
$\{\mathsf{a}_4\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{11}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_8, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{11}, \mathsf{a}_3\}$
$\{\mathsf{a}_4\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{11}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_8, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{11}, \mathsf{a}_3\}$
$\{\mathsf{a}_4\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_8, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{10}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$
$\{\mathsf{a}_4\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$	$\{\mathsf{a}_5, \mathsf{a}_8, \mathsf{a}_7\} \cup \{\mathsf{a}_9\} \cup \{\mathsf{a}_{15}, \mathsf{a}_{16}, \mathsf{a}_{17}, \mathsf{a}_3\}$

Table 2: The sixteen belts of tree A that contain node a_9.

It is easy to see that for every tree node n, there is at least one such belt that contains n: for example, the node set that consists of n itself and of every such leaf that is not a descendant of n is clearly a belt. Therefore, there exists at least one *belt-provider*.

Definition 4. A ***belt-provider*** is any such two-argument function that takes a tree and a node in the tree and returns one such belt of the tree that contains the node.

Definition 5 of a belt-selector, to be presented next, is illustrated by Figure 8: since $\lhd(n_1, n_1') = \langle 3, 1, 4\rangle = \lhd(n_2, n_2')$ and $\lhd(n_1, n_1'') = \langle 3, 1, 2\rangle = \lhd(n_2, n_2'')$, we have $n_1'' \in s(X_1, n_1) \Rightarrow n_2'' \in s(X_2, n_2)$ for any such belt-provider s that is a belt-selector.

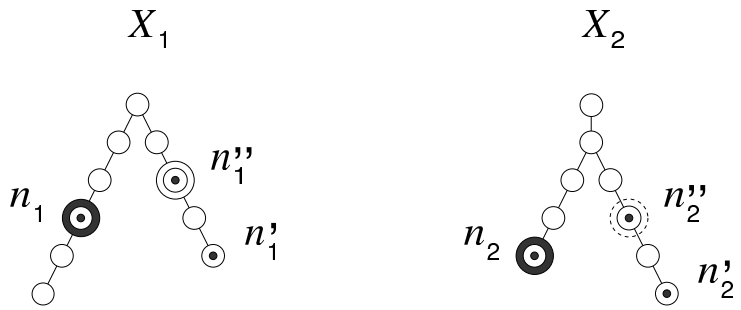

Figure 8: Uniangularity.

Definition 5. A given belt-provider s is ***uniangular***, and hence called a ***belt-selector***, if it meets the following condition.

- Let X_1 and X_2 be two trees. Moreover, let n_1 be a node of X_1, let n_1' be a leaf of X_1, and let the unique ancestor of n_1' in $s(X_1, n_1)$ be denoted as n_1''. Similarly, let n_2 be a node of X_2, let n_2' be a leaf of X_2, and let the unique ancestor of n_2' in $s(X_2, n_2)$ be denoted as n_2''. Then $\lhd(n_1, n_1') = \lhd(n_2, n_2')$ implies $\lhd(n_1, n_1'') = \lhd(n_2, n_2'')$.

The following two examples demonstrate some consequences of the uniangularity requirement.

EXAMPLE 3. ▽

Consider tree B in Figure 9. We argue that there is no such belt-selector that can select belt $\{\mathsf{b}_1, \mathsf{b}_6, \mathsf{b}_3\}$ for node b_3. The reason is that $\lhd(\mathsf{b}_3, \mathsf{b}_5) = \langle 1, -1, 3\rangle = \lhd(\mathsf{b}_3, \mathsf{b}_7)$ but $\lhd(\mathsf{b}_3, \mathsf{b}_1) = \langle 1, -1, 1\rangle \neq \langle 1, -1, 2\rangle = \lhd(\mathsf{b}_3, \mathsf{b}_6)$. △

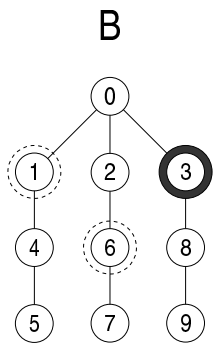

Figure 9: Tree B with a belt that no belt-selector can select for node b_3.

EXAMPLE 4A. ▽

Now we look at four particular belts of tree C, on the left-hand side of

Figure 10, and ask which ones of them may be selected for node c_4 by some belt-selector. (Of course, any belt selected must contain c_4 itself.)

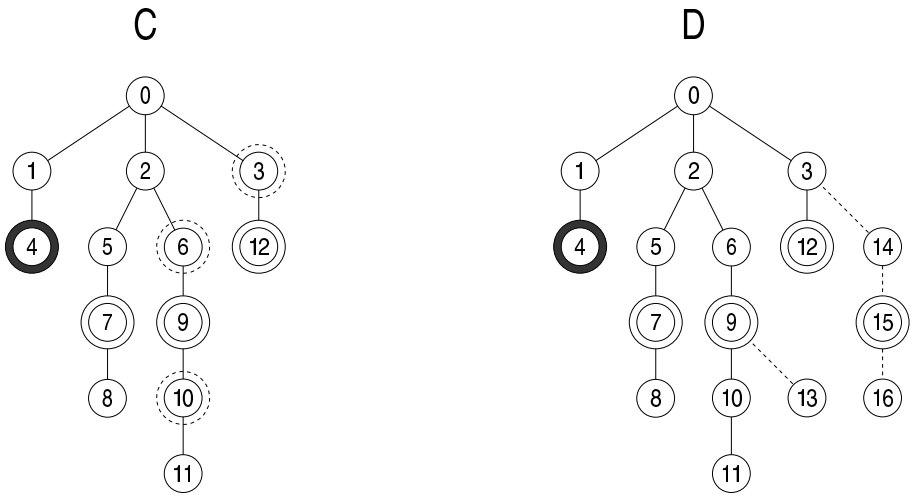

Figure 10: Trees C and D.

1. Consider belt $\{\mathsf{c}_4, \mathsf{c}_7, \mathsf{c}_9, \mathsf{c}_{12}\}$. In Example 4B below, we will argue that there does exist such a belt-selector s for which $s(\mathsf{C}, \mathsf{c}_4)$ equals this belt.
2. Consider $\{\mathsf{c}_4, \mathsf{c}_7, \mathsf{c}_6, \mathsf{c}_{12}\}$, which is the belt of case (1) with the brotherless c_9 replaced by its father c_6. There exists no such belt-selector s^* for which $s^*(\mathsf{C}, \mathsf{c}_4)$ equals this belt.

   Our following proof is by contradiction; suppose for a moment that such s^* exists. Tree D, on the right-hand side of Figure 10, is otherwise fully isomorphic to tree C but has two additional branches, one of which consists of node d_{13} only (and the other of nodes d_{14}, d_{15}, and d_{16}). Because $\lhd(\mathsf{c}_4, \mathsf{c}_8) = \lhd(\mathsf{d}_4, \mathsf{d}_{13})$ and $\lhd(\mathsf{c}_4, \mathsf{c}_7) = \lhd(\mathsf{d}_4, \mathsf{d}_9)$, uniangularity requires that d_9 belongs to $s^*(\mathsf{D}, \mathsf{d}_4)$. (Hence, d_6 in particular cannot belong to $s^*(\mathsf{D}, \mathsf{d}_4)$.) The contradiction desired is now that $\lhd(\mathsf{c}_4, \mathsf{c}_{11}) = \lhd(\mathsf{d}_4, \mathsf{d}_{11})$ but $\lhd(\mathsf{c}_4, \mathsf{c}_6) \neq \lhd(\mathsf{d}_4, \mathsf{d}_9)$, and so uniangularity is violated.
3. Consider $\{\mathsf{c}_4, \mathsf{c}_7, \mathsf{c}_{10}, \mathsf{c}_{12}\}$, which is the belt of case (1) with c_9 replaced by its single son c_{10}. Again, there exists no such belt-selector s^* for which $s^*(\mathsf{C}, \mathsf{c}_4)$ equals this belt.

   The proof is essentially the same as in case (2) above; suppose for a moment that such s^* exists. As in case (2), uniangularity requires that d_9 belongs to $s^*(\mathsf{D}, \mathsf{d}_4)$. (Hence, d_{10} in particular cannot belong to $s^*(\mathsf{D}, \mathsf{d}_4)$.) The contradiction desired is now that $\lhd(\mathsf{c}_4, \mathsf{c}_{11}) = \lhd(\mathsf{d}_4, \mathsf{d}_{11})$ but $\lhd(\mathsf{c}_4, \mathsf{c}_{10}) \neq \lhd(\mathsf{d}_4, \mathsf{d}_9)$, and so uniangularity is violated.
4. Consider $\{\mathsf{c}_4, \mathsf{c}_7, \mathsf{c}_9, \mathsf{c}_3\}$, which is the belt of case (1) with the brotherless c_{12} replaced by its father c_3. Once again, there exists no such belt-selector s^* for which $s^*(\mathsf{C}, \mathsf{c}_4)$ equals this belt.

   Our proof is by contradiction; suppose for a moment that such s^* exists. Consider again tree D. Because $\lhd(\mathsf{c}_4, \mathsf{c}_8) = \lhd(\mathsf{d}_4, \mathsf{d}_{16})$ and $\lhd(\mathsf{c}_4, \mathsf{c}_7) = \lhd(\mathsf{d}_4, \mathsf{d}_{15})$, uniangularity requires that d_{15} belongs to

$s^*(\mathsf{D}, \mathsf{d}_4)$. This forces us to include even d_{12} in $s^*(\mathsf{D}, \mathsf{d}_4)$. The contradiction desired is now that $\lhd(\mathsf{c}_4, \mathsf{c}_{12}) = \lhd(\mathsf{d}_4, \mathsf{d}_{12})$ but obviously $\lhd(\mathsf{c}_4, \mathsf{c}_3) \neq \lhd(\mathsf{d}_4, \mathsf{d}_{12})$, and so uniangularity is violated.

△

2.4 Combs

We let $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}^+$ denote the sets of all integers, all nonnegative integers, and all positive integers, respectively. The 'less-than' relation is extended from $\mathbb{Z}$ to $\mathbb{Z} \cup \{\infty\}$ simply by stating that $t < \infty$ for every $t \in \mathbb{Z}$ and by requiring that the relation remains irreflexive and transitive.

Definition 6. A ***comb*** is any function from $\mathbb{N}^+ \times \{-1, 1\}$ to $\mathbb{N}^+ \cup \{\infty\}$.

Definition 7. A given comb f is a ***characteristic comb*** of a given belt-provider s if for every tree X, for every node n of X, and for every leaf n' of X, the following conditions are met when $\lhd(n, n')$ is denoted as $\langle i, d, j\rangle$ and the unique ancestor of n' that belongs to $s(X, n)$ is denoted as n''.

1. Suppose $d \neq 0$ and $j \leq f(i, d)$. Then $n'' = n'$.
2. Suppose $d \neq 0$ and $j > f(i, d)$. Then n'' is the unique proper ancestor of n' for which $\lhd(n, n'') = \langle i, d, f(i, d)\rangle$.

The following Theorem 8 indicates that the 'is a characteristic comb of' relation is actually a one-to-one correspondence between combs and belt-selectors. In particular, the theorem implies that the set of belt-selectors is non-empty, since the set of combs is obviously non-empty.

Theorem 8. Let the set of belt-providers [respectively, belt-selectors, combs] be denoted as $\mathcal{P}$ [respectively, $\mathcal{S}$, $\mathcal{F}$]. Furthermore, let R denote the set of all such members $\langle f, s\rangle$ of $\mathcal{F} \times \mathcal{P}$ that f is a characteristic comb of s. Then $R \subseteq \mathcal{F} \times \mathcal{S}$, and moreover, R is a bijective function from $\mathcal{F}$ to $\mathcal{S}$.

By Theorem 8, any given belt-selector s has exactly one characteristic comb, and we denote this comb as ϕ_s. (Of course, the theorem also guarantees that $\phi_{s_1} = \phi_{s_2}$ implies $s_1 = s_2$.)

EXAMPLE 4B. ▽

By Theorem 8, there indeed exists a belt-selector realizing case (1) of Example 4A. In fact, we may choose any such belt-selector s that $\phi_s(2, 1) = 3$.

△

Consider what happens when a tree grows by successive leaf unfoldings. Let n and n' be such nodes of a given tree X that $n' \in s(X, n)$ for some belt-selector s. Moreover, suppose that n' is such a leaf that is about to be unfolded. Then by Theorem 8 and Definition 7, the belt selected for n either immediately occupies the new sons of n' or forever remains stuck at n'.

EXAMPLE 5. ▽

Figure 11 shows what happens when a belt-selector s defined as

$$\phi_s(i,d) \;=\; \begin{cases} i & \text{if } d=-1 \\ i+1 & \text{if } d=1 \end{cases}$$

is applied to a node in a growing tree. (Of course, several successive leaf unfoldings take place between each two of the five stages depicted in the figure.) △

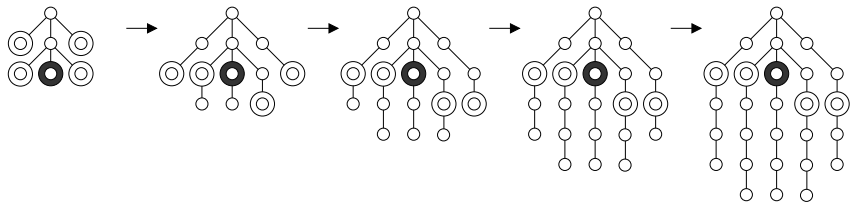

Figure 11: A single belt-selector applied to a node in a growing tree.

2.5 Four important belt-selectors: σ_{E}, σ_{C}, σ_{I}, and $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$

The *extroversive* σ_{E}, the *centroversive* σ_{C}, and the *introversive* σ_{I} are three belt-selectors defined as

$$\begin{aligned} \phi_{\sigma_{\mathsf{E}}}(i,d) &= \infty \\ \phi_{\sigma_{\mathsf{C}}}(i,d) &= i \\ \phi_{\sigma_{\mathsf{I}}}(i,d) &= 1. \end{aligned}$$

The *junction* of a given belt-selector s_1 with a given belt-selector s_2 is denoted as $s_1\|s_2$ and defined as the unique belt-selector for which

$$\begin{aligned} \phi_{s_1\|s_2}(i,-1) &= \phi_{s_1}(i,-1) \\ \phi_{s_1\|s_2}(i,1) &= \phi_{s_2}(i,1). \end{aligned}$$

Figure 12 depicts belt-selectors σ_{E}, σ_{C}, σ_{I}, and $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$. Below, we will use these four as our primary examples of individual belt-selectors.

2.6 On the chosen tree representation

Having now completed the description of belt-selectors, we want to stress that all the previous parts of the present Section 2 have been independent of our actual tree representation, whose details are revealed in this Section 2.6. The choice of the tree representation is not insignificant, for we believe that it strongly affects the approachability of some of the "higher-level" parts of our formal framework. (This concerns especially Theorem 19 in Section 5.1.) Notice, however, that this Section 2.6 is by no means a prerequisite for the rest (that is, Sections 3–7) of this somewhat informal report but does facilitate the absorption of the appendices (that is, Sections A–I).

We impose the following two requirements on the tree representation.

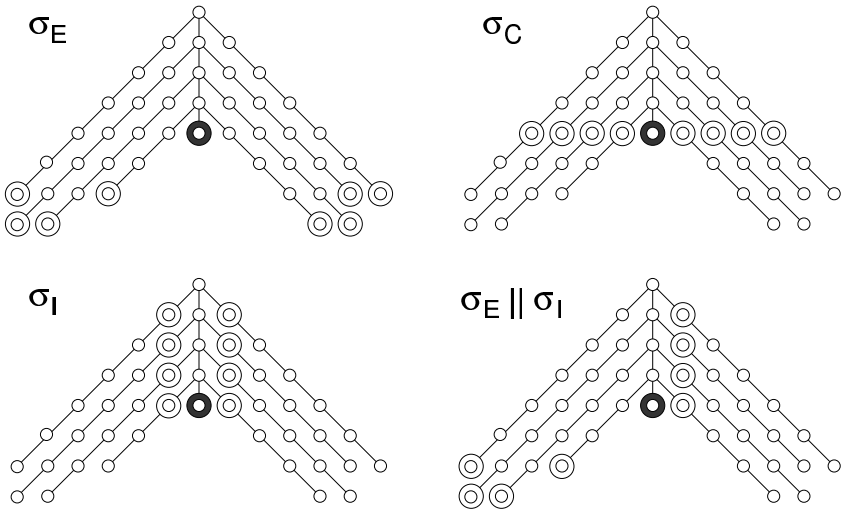

Figure 12: Four important belt-selectors.

- Each tree node should "know" its logical position in the tree. Otherwise, the short notation $\sphericalangle(n_1, n_2)$ for node pair angles is meaningless as no tree is specified. Moreover, we regard the longer alternative '$\sphericalangle(X, n_1, n_2)$' that explicitly fixes the tree as impractical, since a leaf unfolding always changes the tree but never the angle between any two nodes present already in the old tree.
- It should be easy to find out whether a given tree may be expanded into another given tree by successive leaf unfoldings. For us, this is a fundamental relation between two trees.

Our chosen tree representation is as follows. A tree is a finite and nonempty set of nodes. Moreover, each node is provided with an additional component, called a *tag*, that uniquely determines the logical position of the node in the tree. Let us define a unique "base tree" for each tree node recursively as follows: the base tree of the root comprises the root itself; and the base tree of a given non-root node n comprises the base tree of the father of n, n itself, and every brother of n. In our tagging scheme, each node tag encodes the structure of the base tree of the particular node, by fixing both the amount of the base tree nodes and their logical positions.

EXAMPLE 6A. ▽

Consider tree E, on the left-hand side of Figure 13. Tree F, on the right, happens to be isomorphic to the portion of E that constitutes the base tree of node e_7. Furthermore, our tag for e_7 is $[\,\langle 1,2\rangle, \langle 0,0\rangle, \langle 0,1\rangle\,]$, which is to be interpreted as follows.

Each tag is a finite sequence of pairs of nonnegative integers. The length of this particular tag is three, which is the same as the altitude of e_7. The first pair $\langle 1,2\rangle$ indicates that of the sons of the root e_0, one is a left-relative and two are right-relatives of e_7. Similarly, the second pair $\langle 0,0\rangle$ indicates that there are neither left-relatives nor right-relatives of e_7 among the sons of e_2, which is the son of e_0 that is a proper ancestor of e_7. Finally, the third pair $\langle 0,1\rangle$ indicates that the father e_6 of e_7 has exactly one additional son, which

is moreover a right-brother of e_7.

For further concrete tag instances, see Example 6B below. △

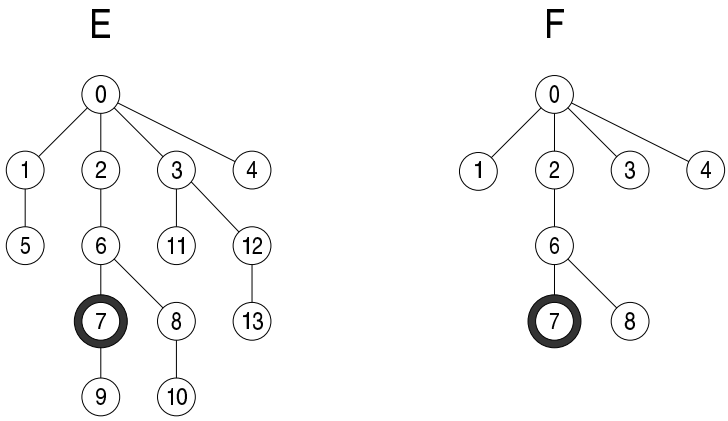

Figure 13: Detecting the base tree of node e_7.

Our claim is that due to the chosen tree representation, a given tree X may be expanded into another given tree X^* by successive leaf unfoldings if and only if $X \subseteq X^*$. This appears rather elegant: the basic 'is a subset of' relation between two sets exactly corresponds to the basic 'may be expanded into, by successive leaf unfoldings' relation between two trees of ours.

Admittedly, our tagging scheme may seem complicated. Its tags do involve some redundant information, whereas a Dewey-type tagging scheme (to be adapted from [6, p. 3]) would have been more economical while still being able to fix the logical tree node positions. The redundancy is necessary, however: since the "asymmetric" Dewey tags ignore every right-relative in the base tree, by adopting them we would have lost the above-mentioned strength of the 'is a subset of' condition.

EXAMPLE 6B. ▽

Table 3 gives both our actual tags and their Dewey counterparts for all the nodes of tree F, on the right-hand side of Figure 13. (This particular Dewey scheme uses 0-based indexing, whereas [6, p. 3] suggests 1-based indexing.) Because the portion of tree E, on the left, that constitutes the base tree of node e_7 is isomorphic to F, each node in this portion has the same two tag "alternatives" as the corresponding node of F. For instance, e_7 has the same tag alternatives as f_7.

Notice the following "non-immunity" of the Dewey scheme: if tag $[1, 1]$, for instance, is added to the Dewey tag set of F, the new set still matches a tree, but not such one that is obtainable from F by successive leaf unfoldings. More specifically, the new tree is otherwise isomorphic to F but contains a single extra leaf, which is a right-brother of the node with the tag of f_6. Furthermore, this non-immunity is not even symmetric: no single extra tag can be added to the Dewey tag set of F to make it match such a tree in which the node with the tag of f_6 has a single extra left-brother. △

n	our tag for n	Dewey tag for n
f_0	$[\,]$	$[\,]$
f_1	$[\langle 0,3\rangle]$	$[0]$
f_2	$[\langle 1,2\rangle]$	$[1]$
f_3	$[\langle 2,1\rangle]$	$[2]$
f_4	$[\langle 3,0\rangle]$	$[3]$
f_6	$[\langle 1,2\rangle, \langle 0,0\rangle]$	$[1,0]$
f_7	$[\langle 1,2\rangle, \langle 0,0\rangle, \langle 0,1\rangle]$	$[1,0,0]$
f_8	$[\langle 1,2\rangle, \langle 0,0\rangle, \langle 1,0\rangle]$	$[1,0,1]$

Table 3: Our tags for tree F and their Dewey counterparts.

2.7 Pointers into the appendices

Our formalization of the notion of a tree, sketched in Section 2.6 above, is called an *expansion tree* and precisely defined in Section B. Each expansion tree node knows its logical position in the tree by Proposition B.17; and Proposition B.30 guarantees that 'is a subset of' and 'may be expanded into, by successive leaf unfoldings' are the same binary relation in the case of expansion trees.

Expansion trees are indeed rooted and directed trees, and our argument is as follows. By part (2) of Proposition B.8, each tree node has at most one father and there is exactly one node without a father. Moreover, the transitive closure of the binary 'is a father of' relation is irreflexive by part (1) of the same proposition. Finally, Proposition B.10 confirms that the sons of any given tree node are totally ordered by the binary 'is a left-brother of' relation.

In the present Section 2, when we speak of a tree and a node, we may implicitly assume that the node belongs to the tree; similarly, when we speak of a node set, we may implicitly assume that there is such a tree that contains the whole set. No such implicit assumptions will be made in Sections A–I.

Section C presents belts and belt-selectors. The definition of the latter is based on the definition of a node pair angle given already in Section B.4.

Combs are not introduced until Section E. The section also demonstrates how conveniently many relations between belt-selectors may be specified by using combs.

3 MONOSYSTEMS

Our abstract model of a string generation tool is called a *monosystem*. The prefix 'mono' stems from the fact that only one belt-selector is involved; the more versatile *trisystems* will be briefly addressed in Section 6.

3.1 Letters and words

We assume that there is a countably infinite set $\mathcal{U}$ of *letters*. A *word* is any finite sequence of letters. The empty word is denoted as Λ, and the set of all words is denoted as $\mathcal{W}$.

Suppose that two instances of a single structured symbol differ from each other by having different values for some attribute of the symbol: for example, two calls of a single macro may have different arguments. The infinity of $\mathcal{U}$ now allows these two symbol instances to be treated as instances of two different letters. Accordingly, on the monosystem level we are able to wholly ignore any internal structure of the symbols we want to process.

We assume that exactly one letter instance is associated with each tree node. (In fact, each tree node is represented simply as a pair consisting of a tag, whose function was described in Section 2.6, and a letter.) Thus each subset of each tree belt specifies a unique word, which is called the *projection* of the belt subset. When a tree leaf is unfolded, in particular, the projection of the unfolding context constitutes the refinement context of the letter instance of the leaf.

EXAMPLE 7A. ▽

Consider the tree in Figure 14. The projection of the belt consisting of the nodes surrounded by (white or black) rings is `gVgTUTVh`. The two subsets of this belt that consist of the left-relatives and of the right-relatives of the leaf with a black ring have projections `gVgT` and `TVh`, respectively. △

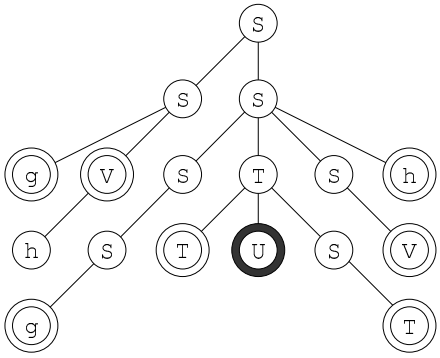

Figure 14: A tree with explicit node letters.

3.2 Letter-refiners

Definition 9. A ***letter-refiner*** is any function from $\mathcal{W} \times \mathcal{U} \times \mathcal{W}$ to $\mathcal{W}$.

Any letter-refiner may be seen as a single-function representation of a set of refinement rules. By Definition 9, the refinement context must be finite but is unbounded in both directions.

EXAMPLE 7B. ▽

Consider such a letter-refiner r that

$$\begin{aligned} r(\mathtt{T},\mathtt{U},\mathtt{T}) &= \mathtt{S} \\ r(\mathtt{gVgT},\mathtt{U},\mathtt{TVh}) &= \mathtt{Vg}. \end{aligned}$$

Suppose that we are to refine the instances of `U` in `TUT` and `gVgTUTVh`. Then r transforms `TUT` into `TST`, and `gVgTUTVh` into `gVgTVgTVh`. Specifically, r does not transform `gVgTUTVh` into `gVgTSTVh`, even if `TUT` is a subword of `gVgTUTVh`. △

Definition 9 above requires that all letter-refiners operate on the same universal letter set $\mathcal{U}$. In practice, however, most letter-refiners *disregard* most members of $\mathcal{U}$. A disregarded letter refines only to itself, cannot be produced from other letters, and is always ignored in the refinement context.

Definition 10. A given letter-refiner r ***disregards*** a given letter c if the following conditions are met for every words w_1 and w_2.

1. $r(w_1, c, w_2) = c$.
2. c does not occur in $r(w_1, c_0, w_2)$ for any such letter c_0 that $c_0 \neq c$.
3. $r(wcw_1, c_0, w_2) = r(ww_1, c_0, w_2)$ and $r(w_1, c_0, w_2cw) = r(w_1, c_0, w_2w)$ for every letter c_0 and for every word w.

3.3 Constituents of a monosystem

We are now ready to define the structure of a monosystem. (The monosystem operation will be described in Section 3.5, which utilizes an important auxiliary concept to be defined in the intermediate Section 3.4.)

Definition 11. A ***monosystem*** is any such quintuple $\langle A, M, c_S, r, s\rangle$ that meets the following conditions.

1. The ***alphabet*** A is a non-empty letter set.
2. M is a non-empty subset of A; the members of M and $A \setminus M$ are called ***mutables*** and ***immutables***, respectively.
3. The ***seed-letter*** c_S is a member of M.
4. r is such a letter-refiner that meets the following subconditions.
   a. r disregards every letter in $\mathcal{U} \setminus A$.
   b. $r(w_1, c, w_2) = c$ for every letter c in $A \setminus M$ and for every words w_1 and w_2.
   c. $r(w_1, c, w_2) \neq \Lambda$ for every letter c and for every words w_1 and w_2.
5. s is a belt-selector.

Concerning the conditions of Definition 11, notice the following.

- Condition (1): the monosystem alphabet may be infinite.

- Conditions (2) and (4b): the immutables may be seen as terminal symbols, since they cannot be further modified by the letter-refiner.
- Condition (3): the rewriting process starts from a letter (that is, from the seed-letter) rather than from a word of arbitrary length.
- Condition (4c): the monosystem letter-refiner cannot replace any letter instance with the empty word Λ. The reason for this restriction is that Λ could not be represented in the history tree. (In practice, Λ must be emulated by some dummy placeholder immutable.)

EXAMPLE 7C. ▽

Some forthcoming examples address such a monosystem $\langle A, M, c_{\mathrm{S}}, r, s\rangle$ whose other constituents than r are as follows.

$$\begin{aligned} A &= \{\mathtt{S},\mathtt{T},\mathtt{U},\mathtt{V},\mathtt{g},\mathtt{h}\} \\ M &= \{\mathtt{S},\mathtt{T},\mathtt{U},\mathtt{V}\} \\ c_{\mathrm{S}} &= \mathtt{S} \\ \phi_s(i,d) &= \begin{cases} 1 & \text{if } i=1 \text{ and } d=-1 \\ \infty & \text{if } i=2 \text{ and } d=-1 \\ 2 & \text{otherwise} \end{cases} \end{aligned}$$

Moreover, we assume that the monosystem letter-refiner r meets the same two constraints as the letter-refiner of Example 7B. (Notice that this specification of r is incomplete.) △

3.4 Belt-selector stagnancy

If a belt-selector is applied to such a tree node at which it is *C-stagnant* for some letter set C, then each such node in the selected belt whose letter belongs to C will remain in the selected belt after any further leaf unfoldings.

Definition 12. Let C be a letter set, and let n be a node in a given tree X. We say that a given belt-selector s is ***C-stagnant*** at n if each node $n' \in s(X,n) \setminus \{n\}$ meets at least one of the following conditions.

1. The letter of n' does not belong to C.
2. $\phi_s(i,d) = j$ when $\lhd(n,n')$ is denoted as $\langle i,d,j\rangle$.

EXAMPLE 7D. ▽

Consider the tree of Example 7A (in Figure 14). For the leaf with the black ring, the belt-selector of the monosystem of Example 7C returns the belt consisting of the nodes with white rings. (Notice that this belt differs from the frontier.) Furthermore, the belt-selector is M-stagnant at this particular leaf when M denotes the mutable set of the monosystem. △

3.5 Operation of a monosystem

A tree leaf that is ready to be unfolded is called *fertile*. The new tree that results from a single leaf unfolding is *directly derivable* from the original tree.

Moreover, any tree that is the result of a series of successive leaf unfoldings is *derivable* from the original tree.

Definition 13. Let n be a tree node, let G be a monosystem, and let the mutable set of G be denoted as M. We say that n is ***fertile*** on G if both the following conditions are met.

1. The letter of n belongs to M.
2. The belt-selector of G is M-stagnant at n.

EXAMPLE 7E. ▽

Consider again the tree of Example 7A (in Figure 14). The leaf with the black ring is the only leaf that is fertile on the monosystem of Example 7C. (In contrast, all the non-leaf nodes happen to be fertile.) △

Definition 14. Let G be a monosystem, and let the letter-refiner and the belt-selector of G be denoted as r and s, respectively. We say that a given tree X' is ***directly derivable*** from a given tree X on G if there are such a node n, such a finite and non-empty node set N, such a letter c, and such words w, w_1, and w_2 that the following conditions are met.

1. n is such a leaf of X that is fertile on G.
2. $X' = X \cup N$, and every node in N is a son of n.
3. c is the letter of n, w is the projection of N, and w_1 [respectively, w_2] is the projection of the subset of $s(X, n)$ that consists of the left-relatives [respectively, right-relatives] of n.
4. $w = r(w_1, c, w_2)$.

EXAMPLE 7F. ▽

The tree in Figure 15 is directly derivable from the tree of Example 7A (in Figure 14) on the monosystem of Example 7C. △

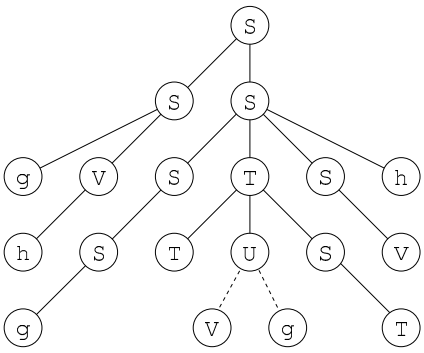

Figure 15: A tree directly derivable from the tree in Figure 14.

We claim that if some tree X' is directly derivable from another tree X, then any fertile node of X is fertile even in X'. (Although even a non-leaf node may technically be fertile, fertileness has any significance only in the case of a leaf.)

Definition 15. For each monosystem G, a binary relation 'is ***derivable*** from, on G' is defined on trees as the reflexive-transitive closure of the 'is directly

derivable from, on G" relation.

We still need to define the set of all *derivatives* of a given monosystem before we are able to state whether a given word is an *export-word*, that is, a word "generated" by the monosystem.

Definition 16. A given tree X is a ***derivative*** of a given monosystem G if X is derivable on G from such a one-node tree whose single node possesses the seed-letter of G.

Definition 17. A given word w is an ***export-word*** of a given monosystem G if both the following conditions are met.

1. Some derivative of G has such a belt whose projection is w.
2. Every letter occurring in w is an immutable of G.

EXAMPLE 8. ▽

Consider a monosystem $\langle A, M, c_{\mathtt{S}}, r, s\rangle$ defined as follows.

$$\begin{aligned} A &= \{\mathtt{S},\mathtt{T},\mathtt{g},\mathtt{h}\} \\ M &= \{\mathtt{S},\mathtt{T}\} \\ c_{\mathtt{S}} &= \mathtt{S} \\ r(w_1,\mathtt{S},w_2) &= \begin{cases} \mathtt{g} & \text{if } \mathtt{T} \text{ occurs in } w_2 \\ \mathtt{ST} & \text{otherwise} \end{cases} \\ r(w_1,\mathtt{T},w_2) &= \begin{cases} \mathtt{hg} & \text{if } \mathtt{S} \text{ occurs in } w_1 \\ \mathtt{T} & \text{otherwise} \end{cases} \\ s &= \sigma_{\mathsf{I}} \end{aligned}$$

Figure 16 shows all the derivatives of the monosystem. It is easy to see that ghg is the only export-word of the monosystem; notice that condition (1) of Definition 17 could be rewritten as "G has such a derivative that the projection of its frontier is w". (The monosystem is actually *confluent*. Confluence will be discussed in Section 5.1, where we claim that every monosystem is confluent.) △

Definition 13 above states that leaves possessing immutables are never fertile. This convention is only a simplification: we claim that the export-word set would not change even if such leaves were treated as always fertile.

3.6 Pointers into the appendices

This Section 3 is matched by the formal Section D, with the following two exceptions. On one hand, tree node letters and tree belt projections are introduced already in Sections B.1 and C.2, respectively. On the other hand, the topic of Section D.6 was touched by a parenthesized remark within Example 8 above but is not properly discussed until Section 5.1 below.

Definition D.6 of a monosystem is a bit more stringent than its counterpart Definition 11 above by requiring that the complement of the monosystem alphabet must be infinite. This additional requirement simply guarantees

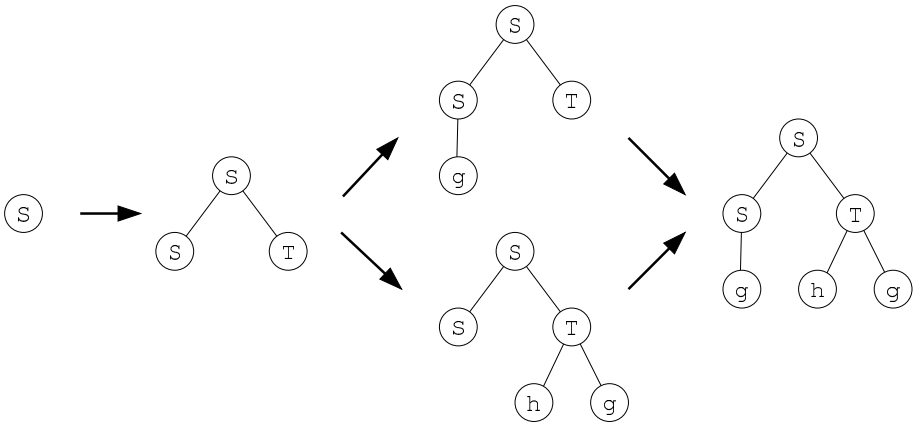

Figure 16: All the derivatives of the monosystem of Example 8.

that new, hitherto unused letters may always be added to the alphabet, which corresponds well with the intended practical applications of the monosystem model.

Definition D.8 of belt-selector stagnancy is equivalent to Definition 12 above but looks different by not employing combs (which are not introduced until Section E); Definition 12 is actually similar to Proposition E.21. In addition, Definitions D.8 and D.11 are more accurate than their respective counterparts Definitions 12 and 13 by explicitly fixing the tree with respect to which stagnancy and fertileness are being defined.

4 EMULATING OTHER DEVICES BY MONOSYSTEMS

We now set out to examine how well the monosystem model is capable of emulating each one of the three program generation tools discussed in Section 1. The term 'emulation' is intentionally left without a definition and thus used in an intuitive sense only. Accordingly, the emulation capability is ignored by the formal presentation constituted by the appendices.

4.1 Macro processors and $\sigma_E||\sigma_I$

We suggest that macro processors may be emulated by using $\sigma_E||\sigma_I$ as the monosystem belt-selector. Macro calls are interpreted as structured symbols, and if two calls of a single macro have different arguments, then they are treated as instances of two different letters. Macro definitions, which constitute the letter-refiner, may freely contain even recursive macro calls.

In addition to the monosystem proper, the emulation nevertheless requires a simple postprocessor that removes all *directives* from each export-word. Every directive is an immutable, and the only such directive that has no exact counterpart already on the macro processor level is the dummy placeholder standing for the empty word, which cannot ever be returned by a monosystem letter-refiner. A representative example of other directives is one that updates the value of some global macro-time variable. This variable may then later be used as an argument of some macro call, or more interestingly, it may steer *conditional macro expansion*.

In the macro processor case, there are two differences between directives and "regular" immutables: first, as suggested above, any directive occurrences should be removed from each export-word; second, the letter-refiner actually ignores all other letters than directives occurring in the refinement context. Therefore, one might ask why we have not made the role of the directive set explicit by adding it as the sixth constituent to the monosystem definition (Definition 11). Our reasons are as follows: the first difference is conceptually a minor one since it concerns only postprocessing; and the second difference is specific to the macro processor case. For example, Section 4.3 below argues that the second difference is nonexistent in the case of the ReFlEx prototype.

4.2 Parametric Lindenmayer systems and σ_C

Monosystem-based emulation of parametric Lindenmayer systems involves two major problems, which originate with the classical Lindenmayer system model.

- Lindenmayer systems have no terminal symbols but the projection of each horizontal belt of the history tree is a valid output string. Provided that each output string is interpreted as a picture-drawing program, each single "run" of a parametric Lindenmayer system produces not only a single picture but a countably infinite series of pictures.
- Lindenmayer systems are often *nondeterministic*: two identical in-

stances of a single symbol may in principle be refined differently even if their refinement contexts are also identical. Hence, two runs of a single nondeterministic parametric Lindenmayer system may produce two different picture series. In contrast, the monosystem letter-refiner is deterministic.

We nevertheless suggest that parametric Lindenmayer systems may, in a restricted sense, be emulated by using σ_C as the monosystem belt-selector. The restriction is that although every picture-drawing program generated by a parametric Lindenmayer system may also be generated by an emulating monosystem, each single emulating monosystem generates only a single program. (See Section 6 on how we may extend the notion of a monosystem to eliminate this restriction.)

A given parametric Lindenmayer system is converted into an emulating monosystem along the following lines. Before the alphabet of the parametric Lindenmayer system is imported to the monosystem level, all the alphabet symbols have to be augmented with such an additional attribute whose value the letter-refiner effectively always increments by one. This makes the alphabets of the node generations (that is, the horizontal belts) of the monosystem-level history tree disjoint: everywhere within a single node generation, the additional attribute has the value specific to that particular generation. Any nondeterminism may now be circumvented, since whenever two instances of a single letter appear within a single monosystem-level history tree, they are possessed by two distinct nodes within a single generation and thus have different refinement contexts. As the final step of the conversion, a maximum value for the additional attribute is chosen, and the corresponding monosystem-level subalphabet is established as the immutable set.

4.3 ReFlEx prototype and σ_I

We suggest that the ReFlEx prototype of a machine-level code generator may be emulated by using σ_I as the monosystem belt-selector. Contrary to the macro processor case, the monosystem-based emulation of ReFlEx scarcely involves directives. The simple reason is that ReFlEx itself scarcely uses directive-like symbols to pass forward information characterizing the refinement context. (Moreover, not a single additional directive needs to be introduced on the monosystem level, since already ReFlEx itself has to use some dummy placeholder for the empty word.)

The context-sensitivity in ReFlEx operation is based on "high-level" built-in domain-specific knowledge rather than on "low-level" directive-like symbols. This means in particular that ReFlEx knows that each symbol it processes is actually a machine-level instruction: the real machine instructions constitute the set of terminal symbols, and nonterminal symbols correspond to macro instructions. ReFlEx furthermore recognizes the registers read and written by each instruction. For example, suppose that register R5 is written by some instruction c_1 and read by another instruction c_2. Moreover, suppose that c is such a macro instruction that does not write R5. Then ReFlEx is smart enough to notice that the instruction sequence replacing the occurrence of c in c_1cc_2 must not use R5 as a temporary data storage.

5 PROPERTIES OF MONOSYSTEMS

We are especially interested in such monosystem properties that depend only on the belt-selector of the monosystem. In other words, we want to examine how the properties of a monosystem reflect the properties of its belt-selector.

5.1 Confluence

Example 8 in Section 3.5 indicated that not all belt-selectors are *rigid* in the sense of Definition 18 immediately below. We assert that both σ_{E} and $\sigma_{\mathsf{E}}||\sigma_{\mathsf{I}}$ are rigid but σ_{C} and σ_{I} are not; moreover, we assert that the belt-selectors of Examples 5 and 7C above are both rigid.

Definition 18. A given belt-selector s is ***rigid*** if every derivative of every such monosystem G whose belt-selector is s has at most one such leaf that is fertile on G.

Even if the belt-selector of a given monosystem happens to be non-rigid, it is fully insignificant which one of the fertile leaves is unfolded first. The reason is that as hinted already by Example 8, every monosystem is *confluent* by Theorem 19, which is depicted in Figure 17. In particular, confluence means that every monosystem has at most one export-word. (We remark that this notion of monosystem confluence is rather strong, as it not only guarantees the existence of such a third tree that is derivable from each one of the two given derivatives but also gives out an effective procedure for the construction of the third tree.)

Theorem 19. Let X_1 and X_2 be two derivatives of a given monosystem G. Then $X_1 \cup X_2$ is such a tree that is derivable both from X_1 and from X_2 on G.

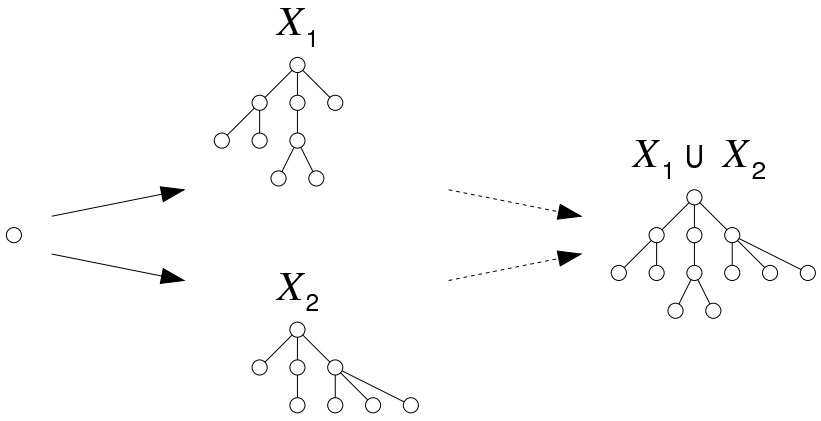

Figure 17: Monosystem confluence.

Theorem 19 clearly assumes that two trees may at least in some cases be combined into one simply by taking the set union. This is indeed possible due to the chosen tree representation described in Section 2.6.

Theorem 19 has an important consequence. Even if the monosystem operation (as specified in Section 3.5) is by definition strictly sequential, in practice the operation of any monosystem with a non-rigid belt-selector may safely be augmented with parallelism. In particular, the combination of the subresults obtained in parallel is utterly simple, as we only need to take the set union. (In Section 5.2.3 below, we argue that certain non-rigid belt-selectors inherently support more parallelism than others.)

5.2 Progressiveness

The more "progressive" the monosystem belt-selector is, the more easily the history tree leaves become fertile. In this Section 5.2, we present a hierarchy of increasing progressiveness for belt-selectors. The hierarchy consists of four main levels, and the fourth and final level moreover comprises a countably infinite subhierarchy. Of each two consecutive hierarchy levels, the latter (that is, the upper) is a non-empty proper subset of the former. Figure 18 already gives an advance sketch of the hierarchy: $\mathcal{S}$ is again the set of all belt-selectors, while 'P' stands for 'progressiveness', 'W' for 'weak', 'S' for 'strong', and 'D' for 'distributive'.

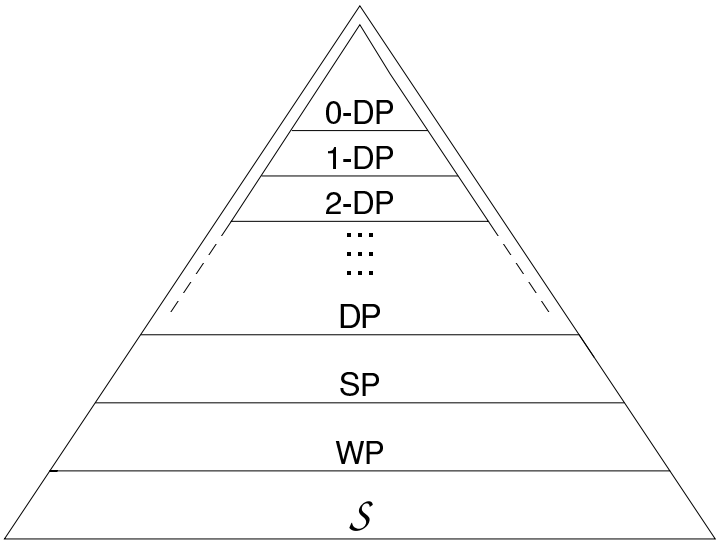

Figure 18: Belt-selector hierarchy of increasing progressiveness.

5.2.1 Weak progressiveness

Weak progressiveness guarantees that the monosystem operation is never blocked unless all the leaves possess immutables.

Definition 20. A given belt-selector s is ***weakly progressive*** if every derivative X of every such monosystem G whose belt-selector is s meets the following condition.

- If X has such a leaf that possesses a mutable of G, then some leaf of X is fertile on G.

Theorem 21. A given belt-selector s is weakly progressive if and only if the following conditions are met for every $\langle i, d\rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

1. If $\phi_s(i, d) > i$, then $\phi_s(i, -d) \leq i$.
2. If $\phi_s(i, d) > i + 1$, then $\phi_s(i', -d) \leq i$ for every $i' > 0$.

We assert that $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$, σ_{C}, and σ_{I} are all weakly progressive but σ_{E} is not; moreover, we assert that the belt-selectors of Examples 5 and 7C are both weakly progressive.

It may be noticed that independently of the notion of a monosystem, every such belt-selector as $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ (or either one of the belt-selectors in Examples 5 and 7C) that is both rigid and weakly progressive uniquely determines a root-to-frontier tree node traversal strategy. More specifically, each rigid and weakly progressive belt-selector s suggests traversing a given tree in the order that the next node to be visited is always the unique such n that every n' meeting the following condition has already been visited: when $\lhd(n, n')$ is denoted as $\langle i, d, j\rangle$, we examine whether $d = 0 \Rightarrow i > 0$ and $d \neq 0 \Rightarrow j < \phi_s(i, d)$. By reversing the above root-to-frontier strategy, we of course obtain a frontier-to-root strategy. (Two different rigid and weakly progressive belt-selectors may well suggest the same traversal strategy. The set of all the possible strategies is actually in one-to-one correspondence with the class of *ideal* belt-selectors to be discussed in Section 5.4.1.)

5.2.2 Strong progressiveness

Whereas weak progressiveness ensures that always at least one of the mutable-lettered leaves is fertile, *strong progressiveness* ensures that each mutable-lettered leaf will eventually become fertile (provided that the choice of the fertile leaf to be unfolded next is fair in the sense that each fertile leaf will eventually become unfolded).

Definition 22. A given belt-selector s is ***strongly progressive*** if every derivative X of every such monosystem G whose belt-selector is s meets the following condition.

- If X has such a leaf n that possesses a mutable of G, then there is such a tree X^* derivable from X on G that n is fertile in X^* on G.

Note that strong progressiveness does imply weak progressiveness.

Theorem 23. A given belt-selector s is strongly progressive if and only if the following conditions are met for every $\langle i, d\rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

1. $\phi_s(i, d) \leq i + 1$.
2. If $\phi_s(i, d) = i + 1$, then $\phi_s(i, -d) \neq i + 1$.

We assert that σ_{C} and σ_{I} are strongly progressive but $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ is not; moreover, we assert that the belt-selector of Example 5 is strongly progressive but the one of Example 7C is not.

5.2.3 Distributive progressiveness

A strongly progressive belt-selector may moreover be *distributively progressive*, which by definition means that it is *k-distributively progressive* for some $k \in \mathbb{N}$. The actual definition of k-distributive progressiveness (which we omit here) implies that if the altitude of each descendant leaf of a given node n is at least k greater than the altitude of n, then at least one of the descendant leaves is fertile (unless they all possess immutables, of course). Hence, the smaller the value of k is, the more fertile leaves there are.

For example, suppose that all the leaves of the derivative in Figure 19 possess mutables, and that the monosystem belt-selector is 3-distributively progressive. Then the derivative has at least four fertile leaves, one in each class of the frontier partition depicted in the figure. (If the belt-selector were 2-distributively progressive, then at least eight fertile leaves would be guaranteed.) This example suggests that no distributively progressive belt-selector is rigid, which is indeed our claim.

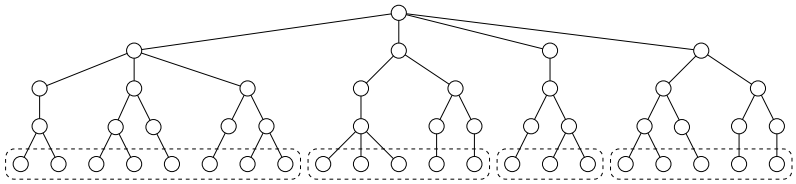

Figure 19: Tree leaves partitioned into four classes.

We assert that σ_{C} is not distributively progressive but σ_{I} is the only 0-distributively progressive belt-selector. Consequently, when the monosystem belt-selector is σ_{I}, every mutable-lettered leaf of every derivative is always fertile.

Non-rigid belt-selectors support parallelism, whose degree varies between highly synchronous and highly asynchronous. A prime example of extremely synchronous parallelism is σ_{C} with its (restricted) capability to emulate parametric Lindenmayer systems. Next, we argue that distributive progressiveness heavily supports extremely asynchronous parallelism.

The operation of a monosystem having a distributively progressive belt-selector may be divided into asynchronously parallel processes as follows. Suppose that a process has been assigned to the task of expanding some branch of the history tree; originally, a single process is assigned to the task of expanding the whole history tree. Distributive progressiveness then implies that eventually (unless immutables are reached before) the process will be able to fork into two or more such subprocesses that expand their respective subbranches and need no later resynchronization whatsoever. Thus, a steadily growing treelike family of asynchronous processes arises. By Theorem 19 above, the final combination of the results of the individual processes is trivial.

5.3 Stableness

Suppose that the letter-refiner of a monosystem is modified so that the re-

finement rule for one mutable is changed. How widely this change may propagate within the history tree depends on the monosystem belt-selector. For example, consider Figure 20, which contains three isomorphic history trees, and suppose that the refinement rule for the mutable of the node with a black dot is changed. The figure shows the possibly affected nodes (which are the black ones) in the cases of belt-selectors $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$, σ_{C}, and σ_{I}.

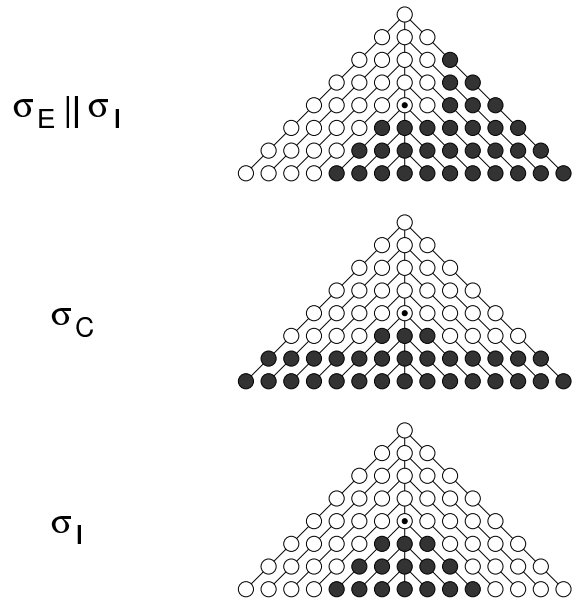

Figure 20: Change propagation with three different belt-selectors.

We say that a belt-selector is *stable* if it is *k-stable* for some $k \in \mathbb{N}$. The definition of k-stableness is as follows: if the change of the refinement rule for the mutable of node n affects node n^*, then $i \leq k$ when $\lhd(n, n^*)$ is denoted as $\langle i, d, j\rangle$. (Notice that $i > 0$ implies that either the refinement rules are not context-free or the belt-selector is not strongly progressive.) Hence, the smaller the value of k is, the smaller is the number of affected nodes. For example, Figure 21 shows the maximal change propagation allowed by 1-stableness, on the left, and by 2-stableness, on the right. (Of course, inward propagation toward the root is never possible.)

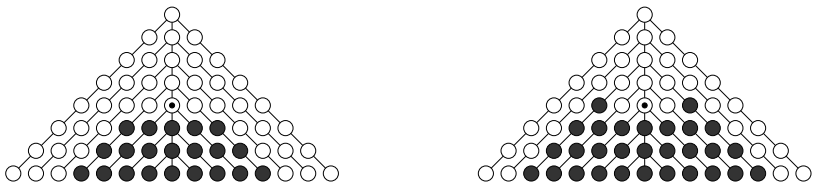

Figure 21: 1-stableness and 2-stableness.

The human writer of the refinement rule base may utilize the possible stableness by making such decisions that are most likely to change in the future near the frontier of the final history tree, rather than near the root.

We claim that a weakly progressive belt-selector is stable if and only if it is distributively progressive. Moreover, we assert that neither $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ nor σ_{C}

is stable, whereas σ_{I} is the only belt-selector that is both 0-stable and weakly progressive. Consequently, when the monosystem belt-selector is σ_{I}, changes in a context-sensitive letter-refiner do not propagate any more than changes in a context-free letter-refiner.

5.4 Soundness

How freely may we select the monosystem belt-selector while still guaranteeing that the possible export-word has the intended semantics? Of course, hardly anything can be guaranteed if the monosystem letter-refiner is such a pathological one that deliberately disrupts the word semantics, and therefore we have to impose some constraints on the letter-refiner. First, in Section 5.4.1 we only require the letter-refiner to be semantics-preserving. Then, in Section 5.4.2 we examine whether the constraints on the belt-selector can be relaxed if the constraints on the letter-refiner are tightened.

5.4.1 Assuming only letter-refiner soundness

Informally, a monosystem is *sound* if the possible export-word has the same semantics as the seed-letter. Moreover, a letter-refiner is *sound* if it always preserves the semantics of the word in which the letter instance to be refined occurs. Finally, a belt-selector is *sound* if every monosystem that combines it with a sound letter-refiner is sound.

Definition 24. A ***semantic classifier*** is any equivalence relation on $\mathcal{W}$.

For any semantic classifier e, we write $w_1 \overset{e}{\sim} w_2$ to state that $\langle w_1, w_2 \rangle \in e$.

Definition 25. Let e be a given semantic classifier. We say that a given monosystem G is ***e*-sound** if $c_{\mathrm{S}} \overset{e}{\sim} w$ for the seed-letter c_{S} of G and for each export-word w of G.

Definition 26. Let e be a given semantic classifier. We say that a given letter-refiner r is ***e*-sound** if $w_1 c w_2 \overset{e}{\sim} w_1 r(w_1, c, w_2) w_2$ for every letter c and for every words w_1 and w_2.

Definition 27. A given belt-selector s is ***sound*** if the following condition is met for every such monosystem G whose belt-selector is s and for every semantic classifier e.

- If the letter-refiner of G is e-sound, then G is e-sound.

Unfortunately, the following Theorem 28 implies that the class of sound belt-selectors is severely restricted.

Theorem 28. Every sound belt-selector is rigid.

Understandably, we are especially interested in such rigid belt-selectors that are not only sound but also weakly progressive. They may be characterized as follows.

Definition 29. A given belt-selector s is ***ideal*** if it is weakly progressive and rigid and if it meets the following further condition.

- $\phi_s(i+1,d) \geq \phi_s(i,d)$ for every $\langle i,d\rangle \in \mathbb{N}^+ \times \{-1,1\}$.

Theorem 30. All the following three statements are equivalent for any weakly progressive belt-selector s.

1. s is sound.
2. s is ideal.
3. The following condition is met for every such monosystem G whose belt-selector is s, for every derivative X of G, and for every leaf n of X: if n is fertile on G, then $s(X,n)$ is the frontier of X.

We assert that $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ is ideal, and therefore also sound, but that σ_{C} and σ_{I} are neither ideal nor sound. Moreover, we assert that the belt-selector of Example 7C is not sound, since it is not ideal even if it is both weakly progressive and rigid. We also claim that in addition to $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$, there are infinitely (even uncountably) many ideal belt-selectors, one of which is the belt-selector of Example 5. The distinguishing feature of $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$ is that with it the letter-refiner may safely assume that mutables never occur on the left-hand side of the refinement context.

5.4.2 Imposing legitimacy constraints on the letter-refiner

Theorem 28 above implies that with a non-rigid belt-selector, it is not enough to use a sound letter-refiner. The letter-refiner has to fulfill an additional *legitimacy* constraint sufficient for the particular belt-selector. Here sufficiency simply means that any given monosystem is guaranteed to be sound if it possesses the particular belt-selector and such a sound letter-refiner that moreover fulfills the legitimacy constraint.

In Section H, we formulate a legitimacy constraint called *semigentleness*. We claim that semigentleness is sufficient for a rather large belt-selector class called *subideal* belt-selectors. For example, we assert that both σ_{C} and σ_{I} are subideal.

Definition 31. A given belt-selector s is ***subideal*** if there are such two ideal belt-selectors s_1 and s_2 that the following conditions are met.

1. $s = s_1\|s_2$.
2. $\phi_{s_1}(i,-1) \leq \phi_{s_2}(i,-1)$ and $\phi_{s_1}(i,1) \geq \phi_{s_2}(i,1)$ for every $i > 0$.

Note that every ideal belt-selector is also subideal. In addition to this fact, Figure 22 sketches how the class of subideal belt-selectors relates to some other belt-selector classes.

Notice especially that the monosystem belt-selector may be safely modified without any need to modify the monosystem letter-refiner as long as the letter-refiner fulfills a legitimacy constraint sufficient even for the new belt-selector. Such a belt-selector modification is able to change the structure but not the semantics of the export-word. For example, if the letter-refiner fulfills the semigentleness constraint, then each one of $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$, σ_{C}, and σ_{I} produces an export-word with the intended semantics (or no export-word at all). Since

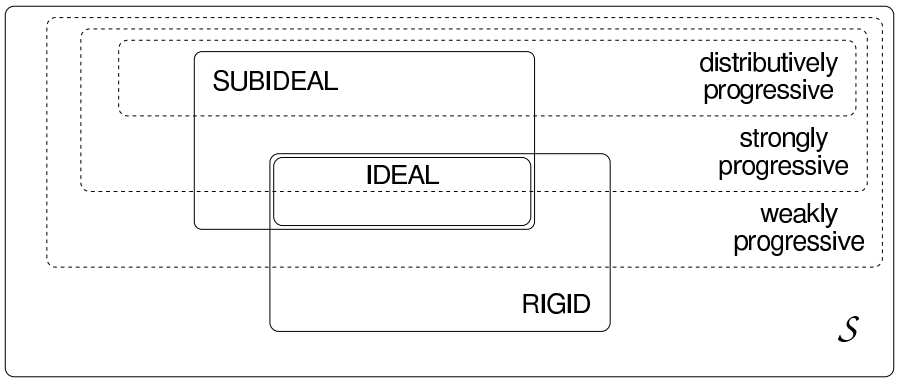

Figure 22: Some belt-selector classes.

the unfolding context given by σ_{I} is the least "greedy" possible, σ_{I} is the one of these three that offers the most parallelism but its export-word typically comes with the least "optimization".

5.5 Coalescence

Macro processing is sensitive to the whole left-hand side of the refinement context (but completely insensitive to the right-hand side) and still feasible in practice. The reason is that the relevant information can be extracted from the refinement context in an incremental fashion. In the monosystem terminology, this means that the letter-refiner r may be implemented by such functions t, t', and t'' that

$$\begin{aligned} r(w_1, c, w_2) &= t'(t(w_1), c) \\ t(w_1 c_1) &= t''(t(w_1), c_1) \end{aligned}$$

for every letters c and c_1 and for every words w_1 and w_2. Moreover, it is essential that both t' and t'' are "easy" to compute. The remaining function $t(w)$ is always computed through t'' and may be seen as the "state" of the macro processor after it has passed text w (which contains only terminal symbols and still includes even the directive-like terminal symbols discussed in Section 4).

In the case of $\sigma_{\mathsf{E}} \| \sigma_{\mathsf{I}}$ (proposed in Section 4 for macro processor emulation), it is indeed advantageous that the letter-refiner admits the suggested decomposition, since the total number of t'' calls required does not exceed the final number of immutable-lettered leaves. This limit is obtained simply as follows: the left-hand side of the unfolding context of any fertile leaf of the history tree contains only immutable-lettered nodes; and whenever two leaves of the history tree are unfolded consecutively, the left-hand side of the refinement context of the former is a prefix of the left-hand side of the refinement context of the latter.

Immediately above, we actually worked out the result that $\sigma_{\mathsf{E}} \| \sigma_{\mathsf{I}}$ is an example of a *left-coalescent* belt-selector, that is, a belt-selector capable of exploiting the suggested letter-refiner decomposition. A more general view of left-coalescence is given by Figure 23. Node n in the tree scheme of the figure belongs to the left-hand unfolding contexts of both node n_1 and node

n_2, of which n_1 happens to be a non-leaf and n_2 a leaf. Let N denote the part of the unfolding context of n_2 that consists of n and its left-relatives. By left-coalescence, the whole N is also included in the unfolding context of n_1. When n_1 was being unfolded, the expansion process perhaps had to laboriously extract from N some relevant information, which is thus available for free now when n_2 is being unfolded.

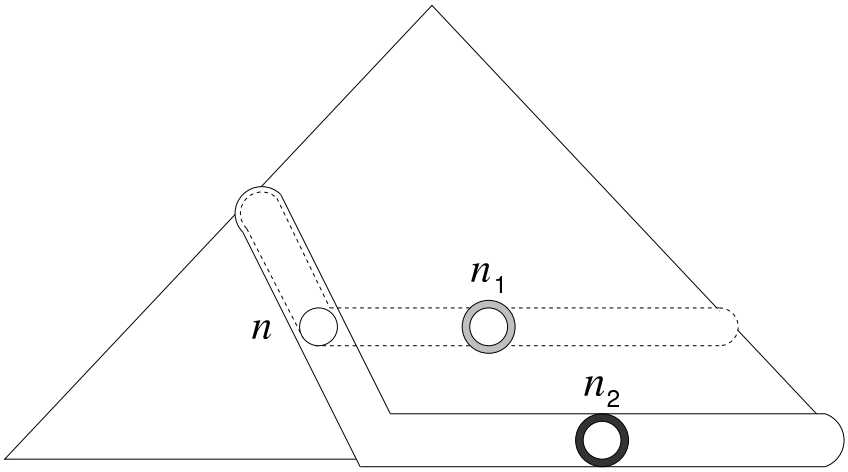

Figure 23: Two tree nodes whose unfolding contexts coalesce.

In the general case, the monosystem letter-refiner is sensitive to both sides of the refinement context. If it moreover admits the appropriate two-sided decomposition, that is,

$$\begin{aligned} r(w_1, c, w_2) &= t'(t_1(w_1), c, t_2(w_2)) \\ t_1(w_1 c_1) &= t_1''(t_1(w_1), c_1) \\ t_2(c_2 w_2) &= t_2''(c_2, t_2(w_2)) \end{aligned}$$

with easy-to-compute t', t_1'', and t_2'', then it is desirable that the belt-selector is *right-coalescent* as well as left-coalescent. We claim that $\sigma_{\mathsf{E}}||\sigma_{\mathsf{I}}$, σ_{C}, and σ_{I} are all both left-coalescent and right-coalescent.

5.6 Pointers into the appendices

Monosystem confluence is dealt with in Section D.6, and belt-selector rigidness in Section F.4. A practical exact condition for rigidness is given by Theorem F.25.

The main emphasis in Section F is on the three types of belt-selector progressiveness. In particular, Definition F.15 characterizes k-distributive progressiveness; exact practical conditions for k-distributive progressiveness and distributive progressiveness are given by Theorems F.20 and F.22, respectively.

Section G considers belt-selector stableness.

Section H deals with soundness, but ideal belt-selectors have been defined already in Section F.6. In Section H.1, the letter-refiner is only required to be sound. Additional legitimacy constraints on the letter-refiner are considered in the remaining Sections H.2—H.6 as follows.

- A mechanism for specifying legitimacy constraints is presented in Section H.2. The legitimacy constraints are specified not directly for letter-refiners but for more general objects called *rewriting maps*. By saying that a legitimacy constraint "fulfilled" by a letter-refiner is "sufficient" for a belt-selector, we actually mean that the belt-selector is *m-wise sound* (as detailed in Definition H.15) for such a rewriting map m that both meets the particular constraint and subsumes (in the sense of Definition H.12) the letter-refiner.
- Section H.3 introduces three simple legitimacy constraints: *reflexivity*, *transitivity*, and *adjunctivity*. By Theorem H.20, adjunctivity is sufficient for σ_{C} but not for σ_{I}, whereas the conjunction of transitivity and adjunctivity is sufficient for σ_{I}.
- Section H.4 introduces three further legitimacy constraints: *civilness*, *semicivilness*, and semigentleness. Semigentleness is weaker than civilness but stronger than semicivilness. By Theorem H.24, semicivilness is similar to adjunctivity by being sufficient for σ_{C} but not for σ_{I}.
- In Section H.5, Theorem H.32 states that civilness is sufficient for the class of *smooth* belt-selectors characterized by Definition H.29.
- Finally, Section H.6 presents the main claim of Section H as Theorem H.39: semigentleness is sufficient for subideal belt-selectors.

The final Section I deals with coalescence. The actual Definition I.3 of left-coalescence is somewhat weaker than the fact we established for $\sigma_{\mathsf{E}}||\sigma_{\mathsf{I}}$ in Section 5.5 above. Consider function t'' that was used in Section 5.5 and whose second argument is a node letter. We argued that with $\sigma_{\mathsf{E}}||\sigma_{\mathsf{I}}$, the number of t'' calls does not exceed the number of immutable-lettered nodes. Actually, left-coalescence only guarantees that there is at most one t'' call per each mutable-lettered node, and so there may be multiple t'' calls per each immutable-lettered node. (Accordingly, it is assumed that the letter possessed by node n in Figure 23 is a mutable.) To put it differently, left-coalescence implies that if the left-hand sides of two unfolding contexts share a mutable-lettered node, then these left-hand sides must unconditionally coalesce; but when the shared node is immutable-lettered, there exists a loophole. It would be possible to eliminate this loophole, but we believe that the definition of left-coalescence would then become unnecessarily narrow for any practical purposes.

6 ON TRISYSTEMS

We propose a *trisystem* as an extension of the monosystem model discussed in Sections 3–5. Trisystems are considered only here and not in the appendices of this report.

6.1 Definition of a trisystem

The control mechanism of a trisystem employs three belt-selectors instead of the single belt-selector of a monosystem. This belt-selector triple is called a *frame*. For the purposes of the present Section 6, we refer to a given frame as $\langle s_{\mathrm{T}}, s_{\mathrm{C}}, s_{\mathrm{E}} \rangle$. The functions of s_{T}, s_{C}, and s_{E} are as follows.

- The *threshold-selector* s_{T} determines leaf fertileness in exactly the same manner as the single belt-selector does on monosystems. Hence, the explicitly addressed belt-selector in part (2) of the trisystem counterpart of Definition 13 would be s_{T}.
- The *context-selector* s_{C} determines the actual unfolding context of a fertile leaf in exactly the same manner as the single belt-selector does on monosystems. Hence, the explicitly addressed belt-selector in part (3) of the trisystem counterpart of Definition 14 would be s_{C}.
- Part (2) of Definition 17 is wholly rejected and the *export-selector* s_{E} now determines the export-words in the manner explained immediately below.

The projection of a given belt B of a given trisystem derivative X is accepted as an export-word if such a new branch can be grafted onto some node of X that both the following conditions are met for some node n^* of the additional branch and for the augmented tree X^* (which need not be a derivative) consisting of X and the additional branch.

- s_{E} is M-stagnant at n^* in X^* when M denotes the mutable set.
- $B = s_{\mathrm{E}}(X^*, n^*) \setminus \{n^*\}$.

For example, suppose that the tree on the left-hand side of Figure 24 is a derivative of such a trisystem whose export-selector is σ_{C}. By the existence of the augmented tree (in which the additional branch consists of a single node) on the right, UUU is an export-word of the trisystem. (The careful reader may notice that due to the node tagging scheme presented in Section 2.6, the tree augmentation may require that some node tags of the original tree are adjusted. In particular, it is certain that at least one node tag in the "output belt" referred to as 'B' above has to be adjusted.)

Note some direct consequences of the particulars of the export-word extraction mechanism. First, the root-only belt never specifies an export-word. Second, two important special cases are rather easily tackled as follows.

- Suppose $s_{\mathrm{E}} = \sigma_{\mathsf{E}}$. Then a given belt specifies an export-word if and only if it is such a frontier whose all node letters are immutables.
- Suppose $s_{\mathrm{E}} = \sigma_{\mathsf{C}}$. Then a given belt B of a given derivative X speci-

Figure 24: Detecting an export-word of such a trisystem whose s_{E} is σ_{C}.

fies an export-word if and only if X contains such a non-root node n that $\sigma_{\mathsf{C}}(X, n) = B$ and σ_{C} is M-stagnant at n for the mutable set M. In particular, every horizontal belt different from the root-only belt therefore specifies an export-word, as suggested already by Figure 24.

By the above, we are in a position to conjecture that the operation of a given monosystem $\langle A, M, c_{\mathrm{S}}, r, s\rangle$ is identical with the operation of the trisystem $\langle A, M, c_{\mathrm{S}}, r, \langle s, s, \sigma_{\mathsf{E}}\rangle\rangle$. In particular, both the derivative set (as well as the binary relations 'is directly derivable from' and 'is derivable from' on trees) and the export-word set are preserved. For example, we are now able to see from Section 4 that macro processors and the ReFlEx prototype may be emulated by frames $\langle \sigma_{\mathsf{E}} \| \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}} \| \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}\rangle$ and $\langle \sigma_{\mathsf{I}}, \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}\rangle$, respectively.

6.2 Properties of trisystems

Similarly to the monosystem case, we are especially interested in such trisystem properties that depend only on the frame of the trisystem.

A frame is said to be *confluent* if every trisystem with it is confluent (in the sense of Theorem 19). We conjecture that not all frames are confluent but a sufficient condition for frame confluence is that $\phi_{s_{\mathrm{T}}}(i, d) \geq \phi_{s_{\mathrm{C}}}(i, d)$ for every $\langle i, d\rangle$. This obviously means that contrary to the monosystem case, not all trisystems are confluent; still, any trisystem with a context-free letter-refiner is confluent.

Frame *soundness* in the trisystem case may be defined analogously to the belt-selector soundness in the monosystem case. We conjecture that in contrast to Theorem 28 above, frame soundness does not require s_{T} to be rigid any more. For example, we suggest that $s_{\mathrm{C}} = s_{\mathrm{E}} = \sigma_{\mathsf{E}}$ guarantees frame soundness, but the particular frame cannot be confluent provided that s_{T} is indeed non-rigid.

Some natural notions of frame equivalence arise. First, two frames are *weakly equivalent* if replacing one with the other cannot change the export-word set of the trisystem. Second, they are *derivationally equivalent* if replacing one with the other cannot change the derivative set of the trisystem. Third and finally, they are *strongly equivalent* if they are both weakly and derivationally equivalent. For example, it should be rather easy to see that the following equivalences hold.

- Frame $\langle \sigma_{\mathsf{E}} || \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}} || \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}} \rangle$, which emulates macro processors, is strongly equivalent to $\langle \sigma_{\mathsf{E}} || \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}, \sigma_{\mathsf{E}} \rangle$.
- Frame $\langle \sigma_{\mathsf{I}}, \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}} \rangle$, which emulates the ReFlEx prototype, is weakly but not derivationally equivalent to $\langle \sigma_{\mathsf{E}} || \sigma_{\mathsf{I}}, \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}} \rangle$.
- Any two frames are derivationally equivalent if they have both the same s_{T} and the same s_{C}.

6.3 Emulating other devices by nondeterministic trisystems

For intuitively satisfactory emulation of Chomsky grammars [27, 7, 16] and classical Lindenmayer systems, which are the two most famous abstract generative devices of formal language theory, we have to further relax the trisystem definition by allowing the letter-refiner to be nondeterministic. Instead of a single word to be used as the replacement, the letter-refiner should return a finite and non-empty word set to which the actual replacement must belong. The finite refinement rule set of a context-sensitive (that is, type-1) Chomsky grammar [16, pp. 29–30] or a classical Lindenmayer system may then be represented as a single nondeterministic letter-refiner (which in this case is effectively sensitive only to a bounded refinement context and effectively operates on a finite alphabet). Of course, this introduction of nondeterminism means that frame confluence in the strictest sense becomes impossible.

A distinctive property of such a nondeterministic letter-refiner r that emulates the finite rule set of a context-sensitive Chomsky grammar is that $r(w_1, c, w_2) \subseteq r(w_1' w_1, c, w_2 w_2')$ for every letter c and every words w_1, w_2, w_1', and w_2'. In contrast, [25, p. 281] indicates that this property does not carry over to the emulation of classical Lindenmayer systems.

6.3.1 Chomsky grammars and frame $\langle \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}, \sigma_{\mathsf{E}} \rangle$

We suggest that context-sensitive Chomsky grammars may be emulated by frame $\langle \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}, \sigma_{\mathsf{E}} \rangle$. Since $s_{\mathrm{T}} = \sigma_{\mathsf{I}}$, every mutable-lettered leaf is always fertile; and since $s_{\mathrm{C}} = \sigma_{\mathsf{E}}$, the leaf unfolding context always equals the frontier. However, we have to set two additional restrictions on the rule set of the context-sensitive Chomsky grammar.

- In general, a context-sensitive Chomsky grammar may contain a single length-reducing rule of the form $w_1 c w_2 \to w_1 w_2$ (see [27, p. 83] for more on this anomaly), that is, $c_0 \to \Lambda$ for the start symbol c_0. We forbid this rule, because the word set returned by the trisystem letter-refiner must not contain the empty word. This simply means that for a given context-sensitive language L, we are only able to generate the similarly context-sensitive $L \setminus \{\Lambda\}$.
- There should be at least one rule of the context-free form $c \to w$ for each nonterminal c, because the word set returned by the trisystem letter-refiner must never be empty. This restriction has no effect on the set of languages that can be generated, since we may always choose $w = c$.

We also suggest that this particular frame $\langle \sigma_{\mathsf{I}}, \sigma_{\mathsf{E}}, \sigma_{\mathsf{E}} \rangle$ is sound. (The introduction of nondeterminism does not affect frame soundness, provided that the definition of letter-refiner soundness is adjusted in the reasonable way.)

Finally, let us consider relaxing the emulation scheme by permitting the use of a dummy placeholder immutable for representing the empty word Λ. Then by the above, any context-sensitive Chomsky grammar may trivially be converted into such a nondeterministic trisystem that generates the same language. More interestingly, [15, theorem 1.4] implies that any recursively enumerable (that is, type-0) language can be generated by such a Chomsky grammar whose every rule is either of the form $w_1 c w_2 \to w_1 c' w' w_2$ or of the form $c \to \Lambda$. Because of the relaxation of the empty word representation, even this Chomsky grammar may now trivially be converted into a nondeterministic trisystem.

6.3.2 Lindenmayer systems and frame $\langle \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}} \rangle$

We suggest that a variety of classical Lindenmayer systems may be emulated by frame $\langle \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}} \rangle$. (More specifically, any such FPIL system [9, pp. 274–275] whose start words do not include Λ can be coped with. Furthermore, any FIL system can be coped with if it is permitted to use a dummy placeholder letter for representing Λ.) This frame also emulates parametric Lindenmayer systems elegantly, whereas $\langle \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}}, \sigma_{\mathsf{E}} \rangle$ corresponds to the clumsy emulation by monosystems suggested in Section 4.

We also suggest that even if this particular frame $\langle \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}}, \sigma_{\mathsf{C}} \rangle$ is not sound, either one of adjunctivity and semicivilness mentioned in Section 5.6 is a sufficiently strong legitimacy constraint for it.

Since the Lindenmayer rewriting starts from a word rather than from a letter, a single extra letter (in addition to the possible empty word placeholder) to be used as the seed-letter has to be included in the trisystem alphabet. Because the root-only belt of any trisystem derivative never specifies an export-word, the extra letter does not occur in any export-word.

A Lindenmayer system may be emulated by a trisystem that has no immutables, but a monosystem without immutables cannot have any export-word. Accordingly, the distinction between weak and strong progressiveness of s_{T} is highly significant in the trisystem case: a strongly progressive s_{T} guarantees that every branch of the history tree grows beyond any predefined limit provided that the immutable set is empty. In contrast, in the monosystem case the distinction between these two progressiveness types has little significance, due to the particulars of the inflexible export-word extraction mechanism.

In conclusion, we feel that the basic principles of the parallel Lindenmayer rewriting are satisfactorily captured by the trisystem model, even if the trisystem operation is at least by definition strictly sequential.

7 CONCLUSION

We have presented a monosystem as an abstract generative string rewriting device aimed at modeling such practical program generation and optimization tools as macro processors and parametric Lindenmayer systems. Monosystems differ from such classical abstract generative devices of formal language theory as Chomsky grammars by being able to operate on an infinite alphabet and by being sensitive to an unbounded two-sided context. The infinite alphabet makes it conceptually simple to deal with structured symbols (such as macro calls), and the unbounded context-sensitivity furthers optimization. We believe that these two relaxations make monosystems fit for program generation: even if Chomsky grammars are traditionally specified as generative devices rather than as recognizer devices (see [27, pp. 9–11] for a simple reversal procedure giving the recognizer specification), their primary application area has been program analysis rather than program generation. Nevertheless, the trisystem extension of the basic monosystem model is capable of emulating both type-1 Chomsky grammars and a variety of classical Lindenmayer systems.

We suggest that a practical monosystem-based tool should usually have a letter-refiner that fulfills a legitimacy constraint such as semigentleness mentioned in Section 5.4.2. This scheme incorporates the following favorable features.

- By tuning the belt-selector, we may opt for sequential, synchronously parallel, or asynchronous parallel rewriting. For example, these three forms of rewriting follow from the use of the ideal $\sigma_{\mathsf{E}}\|\sigma_{\mathsf{I}}$, the nonrigid and strongly progressive σ_{C}, and the distributively progressive σ_{I}, respectively. (As indicated in Section 5.4.2, each one of these belt-selectors is subideal, and hence semigentleness is a sufficiently strong legitimacy constraint for them all.) In particular, the 0-distributively progressive σ_{I} allows any such two nodes that are not an ancestor-descendant couple to be unfolded by two distinct asynchronously parallel processes.
- The monosystem remains sound independently of the degree of parallelism. Changing the belt-selector may change the structure but not the semantics of the output, as long as the legitimacy constraint is sufficiently strong even for the new belt-selector.
- Independently of the strength of the legitimacy constraint, parallelism is promoted by monosystem confluence expressed in Theorem 19: the results of any processes executed in parallel may always be safely and effortlessly combined into one.

Immediately above, we essentially advocated that the letter-refiner of a practical monosystem-based tool should provide built-in support for parallelism. Using parallelism however involves some trade-offs. On one hand, the advantages of parallelism may be recalled as follows.

- Parallel rewriting is in principle faster since it may be implemented by a multiprocessor network. This holds in particular in the asyn-

chronous case of distributively progressive belt-selectors: by "asynchronousness" we always mean that no interprocess communication or synchronization whatsoever is needed.

- Asynchronous parallelism (or rather, distributive progressiveness of the belt-selector) results in stableness in the sense of Section 5.3.

On the other hand, the disadvantages of parallelism may be summarized as follows.

- In addition to the very natural soundness requirement, the monosystem letter-refiner indeed has to fulfill a further legitimacy constraint and may thus be prevented from performing some optimizations.
- Even if the monosystem letter-refiner supports parallelism (by fulfilling a sufficiently strong legitimacy constraint), the output with the most optimization is typically still obtained when the particular letter-refiner is driven strictly sequentially by some ideal belt-selector.

References

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[2] P. J. Brown. A survey of macro processors. *Annual Review in Automatic Programming*, vol. 6, part 2, pp. 37–88. Pergamon Press, 1969.

[3] P. J. Brown. *Macro Processors and Techniques for Portable Software*. Wiley, 1974.

[4] M. Campbell-Kelly. *An Introduction to Macros*. Macdonald, London (UK), 1973.

[5] A. J. Cole. *Macro Processors* (second edition). Cambridge University Press, 1981.

[6] F. Gécseg and M. Steinby. Tree languages. In [26], vol. 3, pp. 1–68.

[7] M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

[8] G. T. Herman and G. Rozenberg. *Developmental Systems and Languages*. North-Holland, 1975.

[9] L. Kari, G. Rozenberg, and A. Salomaa. L systems. In [26], vol. 1, pp. 253–328.

[10] E. Lassila. ReFlEx—an experimental tool for special-purpose processor code generation. Report B15, Digital Systems Laboratory, Helsinki University of Technology. Espoo (Finland), March 1996.

[11] E. Lassila. A macro expansion approach to embedded processor code generation. *Proceedings of the 22nd EUROMICRO Conference*, pp. 136–142. IEEE Computer Society Press, 1996.

[12] E. Lassila. Towards optimizing code generation by domain-sensitive macro expansion. Report A42, Digital Systems Laboratory, Helsinki University of Technology. Espoo (Finland), January 1997.

[13] E. Lassila. On tree belts and belt-selectors. In N. Husberg, T. Janhunen, and I. Niemelä (eds.), *Leksa Notes in Computer Science: Festschrift in Honour of Professor Leo Ojala*, pp. 47–58. Report HUT-TCS-A63, Laboratory for Theoretical Computer Science, Helsinki University of Technology. Espoo (Finland), October 2000.

[14] P. Marwedel and G. Goossens (eds.). *Code Generation for Embedded Processors*. Kluwer, 1995.

[15] A. Mateescu and A. Salomaa. Aspects of classical language theory. In [26], vol. 1, pp. 175–251.

[16] A. Mateescu and A. Salomaa. Formal languages: an introduction and a synopsis. In [26], vol. 1, pp. 1–39.

[17] R. Měch. Modeling and simulation of the interaction of plants with the environment using L-systems and their extensions. Ph. D. thesis, Department of Computer Science, The University of Calgary. Calgary (Alberta, Canada), November 1997.

[18] R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. *Proceedings of SIGGRAPH 96 Conference*, pp. 397–410. ACM SIGGRAPH, 1996.

[19] R. Morgan. *Building an Optimizing Compiler*. Butterworth-Heinemann, 1998.

[20] S. S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1997.

[21] P. Prusinkiewicz. Simulation modeling of plants and plant ecosystems. *Communications of the ACM*, vol. 43, no. 7, pp. 84–93. 2000.

[22] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Měch. Visual models of plant development. In [26], vol. 3, pp. 535–597.

[23] P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. *Proceedings of SIGGRAPH 94 Conference*, pp. 351–358. ACM SIGGRAPH, 1994.

[24] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1990.

[25] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, 1980.

[26] G. Rozenberg and A. Salomaa (eds.). *Handbook of Formal Languages* (vols. 1–3). Springer, 1997.

[27] A. Salomaa. *Formal Languages*. Academic Press, 1973.

A MATHEMATICAL PRELIMINARIES

A.1 General on notation and terminology

A.1.1 Sets

We let $\emptyset$, $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}^+$ denote the empty set, the set $\{\ldots, -1, 0, 1, \ldots\}$ of all integers, the set $\{0, 1, \ldots\}$ of all nonnegative integers, and the set $\{1, 2, \ldots\}$ of all positive integers, respectively.

Let T be a set. We write $t \in T$ or $t \notin T$ if t is or is not a member of T, respectively. By saying "T contains t", we mean that $t \in T$; but by saying "T consists of t", we mean that $T = \{t\}$. We write $T_0 \subseteq T$ to state that T_0 is a *subset* of T, or equivalently, that T is a *superset* of T_0. The set of all subsets of T is denoted as 2^T.

Let T_1 and T_2 be sets. The *union* and *intersection* of T_1 and T_2 are denoted as $T_1 \cup T_2$ and $T_1 \cap T_2$, respectively. (Rather than only to two sets, the union and intersection operations may be applied to any non-zero number of sets.) The set that consists of every such member of T_1 that is not a member of T_2 is denoted as $T_1 \setminus T_2$.

A.1.2 Cartesian products, binary relations, and functions

The *Cartesian product* of two sets T_0 and T_1 is denoted as $T_0 \times T_1$. If $t_0 \in T_0$ and $t_1 \in T_1$, then $\langle t_0, t_1 \rangle$ denotes a particular member of $T_0 \times T_1$. For any two sets, the members of their Cartesian product are called *pairs*. Accordingly, by "a pair" we always mean an ordered pair, as the Cartesian product is not commutative.

The Cartesian product is associative. Suppose we have sets $T_0, \ldots, T_{k-1}$ for some positive integer k, and suppose again that $t_i \in T_i$ for every such i that $0 \leq i < k$. Then the Cartesian product of these sets is denoted as $T_0 \times \cdots \times T_{k-1}$, and a particular one of its members as $\langle t_0, \ldots, t_{k-1} \rangle$. Furthermore, if all the sets T_i are equal to some set T, then this Cartesian product may be denoted shortly as T^k. The members of the Cartesian product of k given sets, with $k \geq 1$, are called *k-tuples*. A k-tuple may optionally be called a pair, a *triple*, a *quadruple*, a *quintuple*, or a *sextuple*, depending on the exact value of k.

A *binary relation* on a given set T is any subset of $T \times T$. When R is a binary relation on T, and when T' is a subset of T, we say that the *restriction* of R to T' is the binary relation on T' that equals $R \cap (T' \times T')$.

A *function* f from a set T to a set U, which may be denoted as $f : T \to U$, is any such subset of $T \times U$ that meets the following condition: for every $t \in T$ there is exactly one such $u \in U$ that $\langle t, u \rangle \in f$; this unique u may be denoted as $f(t)$. The two sets T and U are called the *domain* and the *codomain* of f, respectively. The subset of U that consists of every such $u \in U$ that $f(t) = u$ for at least one $t \in T$ is called the *image* of f.

Let $f : T \to U$ be a function. We say that f is *surjective* if its image is U, and we say that f is *injective* if $f(t_1) = f(t_2) \Rightarrow t_1 = t_2$ for every t_1 and t_2 in T. Finally, we say that f is *bijective* if it is both surjective and injective.

A.1.3 Finite sequences

A *finite sequence* z is technically a function whose domain is such a finite subset $\{0, \ldots, k-1\}$ of $\mathbb{N}$ that consists of the k smallest nonnegative integers for some $k \in \mathbb{N}$. This k is called the *length* of z and denoted as $|z|$. We say that z is *empty* if $|z| = 0$. We may give a simple representation for z by listing its values as $[\, z(0), \ldots, z(|z|-1) \,]$; thus, z is empty if and only if $z = [\,]$.

Let two given finite sequences $[\, t'_0, \ldots, t'_{k'-1} \,]$ and $[\, t''_0, \ldots, t''_{k''-1} \,]$ be denoted as z' and z'', respectively. The *catenation* of z' and z'' is denoted as $z' \oplus z''$ and defined as the finite sequence $[\, t'_0, \ldots, t'_{k'-1}, t''_0, \ldots, t''_{k''-1} \,]$, whose length is the sum of the lengths of z' and z''. It is obvious that the catenation operation is associative but not commutative.

A finite sequence z_0 is a *prefix* of a finite sequence z if there is such a finite sequence z^* that $z = z_0 \oplus z^*$. Analogously, z_0 is a *suffix* of z if there is such a finite sequence z' that $z = z' \oplus z_0$.

A.2 Properties of binary relations

Definition A.1. Let R be a binary relation on a set T.

1. R is ***reflexive*** if $\langle t, t \rangle \in R$ for every $t \in T$.
2. R is ***irreflexive*** if $\langle t, t \rangle \notin R$ for every $t \in T$.
3. R is ***symmetric*** if $\langle t, t' \rangle \in R \Rightarrow \langle t', t \rangle \in R$ for every pair $\langle t, t' \rangle \in T^2$.
4. R is ***antisymmetric*** if $\{\langle t, t' \rangle, \langle t', t \rangle\} \subseteq R \Rightarrow t = t'$ for every pair $\langle t, t' \rangle \in T^2$.
5. R is ***transitive*** if $\{\langle t, t' \rangle, \langle t', t'' \rangle\} \subseteq R \Rightarrow \langle t, t'' \rangle \in R$ for every triple $\langle t, t', t'' \rangle \in T^3$.

Definition A.2. Let R be a binary relation on a set T. The ***reflexive closure*** [respectively, ***transitive closure***, ***reflexive-transitive closure***] of R is the binary relation on T that is the intersection of every such binary relation on T that is reflexive [respectively, transitive, both reflexive and transitive] and a superset of R.

Definition A.3. Let R be a binary relation on a set T.

1. R is an ***equivalence*** if it is reflexive, symmetric, and transitive.
2. R is a ***partial order*** if it is reflexive, antisymmetric, and transitive.
3. R is a ***partial strict-order*** if it is irreflexive, antisymmetric, and transitive.
4. R is a ***total order*** [respectively, a ***total strict-order***] if it is a partial order [respectively, a partial strict-order] and if for every such $\langle t, t' \rangle \in T^2$ that $t \neq t'$, it is the case that $\langle t, t' \rangle \in R$ or $\langle t', t \rangle \in R$.

Proposition A.4. A given binary relation on a given set is a partial strict-order if and only if it is both irreflexive and transitive.

Proposition A.5. Let T be such a set whose all members are themselves sets. Then the binary 'is a subset of' relation on T is a partial order.

A.3 Letters and words

Specification A.6. There is a countably infinite set $\mathcal{U}$ of ***letters***.

Definition A.7. For any letter set C, the ***complement*** of C is denoted as $\neg C$ and defined as the set $\mathcal{U} \setminus C$.

Proposition A.8. $\neg(\neg C) = C$ for any letter set C.

Definition A.9. A ***word*** is any finite sequence of letters. The set of all words is denoted as $\mathcal{W}$.

Notation A.10. The empty word may be denoted as Λ; and any other word $[c_0, \ldots, c_{k-1}]$ may be written simply as $c_0 \cdots c_{k-1}$. In particular, if c is a letter, then the expression 'c' may also be interpreted as a word such that $|c| = 1$. Moreover, the word $w_1 \oplus w_2$ may be written simply as $w_1 w_2$ for any two words w_1 and w_2; and even the catenation of more than two words may be written equally simply, such as $w_1 w_2 w_3$, by the associativity of catenation of finite sequences.

Proposition A.11. $\mathcal{W}$ is a countably infinite set.

B EXPANSION TREES

B.1 Structure of an expansion tree

► An expansion tree consists of a finite and non-zero number of nodes. The concrete representation of an expansion tree is simply a node set; each node in the set is, however, accompanied by a tag, which uniquely determines the logical position of the node in the tree (as guaranteed by Proposition B.17). Moreover, the tags are chosen in such a way that a tree is a subset of another one if and only if the latter can be obtained from the former by a series of successive leaf unfoldings (as guaranteed by Proposition B.30).

Definition B.1. A ***tag*** is any finite sequence of pairs of nonnegative integers.

Definition B.2. An ***expansion node***, or simply a ***node***, is any such pair $n = \langle y, c \rangle$ that y is a tag and c is a letter. Moreover, these y and c may be denoted as $\tau(n)$ and $\alpha(n)$, respectively.

Definition B.3. An ***expansion tree***, or simply a ***tree***, is any such finite and non-empty set X of nodes that meets the following conditions.

1. Each two distinct nodes in X possess distinct tags.
2. Let y be a tag, and let y' be a prefix of y. Now if X contains a node whose tag is y, then X also contains a node whose tag is y'.
3. Let y be a tag, and let p_1, p_2, q_1, and q_2 be nonnegative integers. Moreover, suppose that X contains a node whose tag is $y \oplus [\langle p_1, q_1 \rangle]$. Then $p_1 + q_1 = p_2 + q_2$ if and only if X contains a node whose tag is $y \oplus [\langle p_2, q_2 \rangle]$.

Notation B.4. The set of all nodes is denoted as $\mathcal{N}$, and the set of all trees is denoted as $\mathcal{X}$.

Proposition B.5. Suppose that a tree X contains a node n. Then $(X \setminus \{n\}) \cup \{\langle \tau(n), c \rangle\}$ is a tree for any letter c.

Proposition B.6. For any node n, there is such a tree that contains n.

B.2 Basic relations between tree nodes

Definition B.7. A node n_1 is a ***father*** of a node n_2 if $\tau(n_1)$ is such a prefix of $\tau(n_2)$ that $|\tau(n_1)| = |\tau(n_2)| - 1$.

Proposition B.8. These statements hold.

1. The transitive closure of the 'is a father of' relation is irreflexive.
2. For any tree X, it is the case that each node of X has at most one father in X, and there is exactly one such node in X that has no father in X.

Definition B.9. A node n_1 is a ***left-brother*** of a node n_2 if there are such a tag y and such nonnegative integers p_1, p_2, q_1 and q_2 that meet the following conditions.

1. $p_1 < p_2$ and $p_1 + q_1 = p_2 + q_2$.

2. $\tau(n_1) = y \oplus [\langle p_1, q_1 \rangle]$.
3. $\tau(n_2) = y \oplus [\langle p_2, q_2 \rangle]$.

Proposition B.10. These statements hold.

1. The 'is a left-brother of' relation is a partial strict-order.
2. The following two substatements are equivalent for every two distinct nodes n_1 and n_2 in any single tree X.
   a. One of n_1 and n_2 is a left-brother of the other.
   b. X contains such a node that is a father of both n_1 and n_2.

B.3 More relations between tree nodes

Definition B.11. Let n_1 and n_2 be two nodes.

1. n_1 is a ***proper ancestor*** of n_2 if the pair $\langle n_1, n_2 \rangle$ belongs to the transitive closure of the 'is a father of' relation.
2. n_1 is an ***ancestor*** of n_2 if $\langle n_1, n_2 \rangle$ belongs to the reflexive-transitive closure of the 'is a father of' relation.
3. n_1 is a ***left-relative*** of n_2 if there are such nodes n_1^* and n_2^* that meet the following conditions.
   a. n_1^* is a left-brother of n_2^*.
   b. n_1^* is an ancestor of n_1.
   c. n_2^* is an ancestor of n_2.

Proposition B.12. The 'is an ancestor of' relation is a partial order, whereas both the 'is a proper ancestor of' and the 'is a left-relative of' relations are partial strict-orders.

Definition B.13. Let X be a tree, and let n be a node.

1. n is a ***root*** of X if $n \in X$ and n is an ancestor of every node in X.
2. n is a ***leaf*** of X if $n \in X$ and n is a proper ancestor of no node in X.
3. The ***frontier*** of X is the set of all leaves of X.

Proposition B.14. Every tree has exactly one root and at least one leaf.

Definition B.15. Let n_1 and n_2 be two nodes.

1. n_1 is a ***son*** [respectively, a ***proper descendant***, a ***descendant***] of n_2 if n_2 is a father [respectively, a proper ancestor, an ancestor] of n_1.
2. n_1 is a ***right-brother*** [respectively, a ***right-relative***] of n_2 if n_2 is a left-brother [respectively, a left-relative] of n_1.

Proposition B.16. For any two nodes n_1 and n_2 in any single tree, exactly one of the following statements hold.

1. $n_1 = n_2$.
2. n_1 is a proper ancestor of n_2.
3. n_1 is a proper descendant of n_2.
4. n_1 is a left-relative of n_2.
5. n_1 is a right-relative of n_2.

Proposition B.17. Let n_1 and n_2 be nodes of given trees X_1 and X_2, re-

spectively. Then $\tau(n_1) = \tau(n_2)$ if and only if the following conditions are met.

1. n_1 has exactly as many ancestors in X_1 as n_2 has in X_2.
2. Let n_1^* be such a node of X_1 that is an ancestor of n_1, and let n_2^* be such a node of X_2 that is an ancestor of n_2. Moreover, suppose that n_1^* has exactly as many ancestors in X_1 as n_2^* has in X_2. Then n_1^* has both exactly as many left-brothers and exactly as many right-brothers in X_1 as n_2^* has in X_2.

B.4 Node altitudes and node pair angles

Definition B.18. A nonnegative integer k is an ***altitude*** of a node n if there is such a tree that contains both n and exactly k proper ancestors of n.

Proposition B.19. Each node has exactly one altitude, and this unique altitude is the same as the length of the tag of the node.

Definition B.20. An integer triple $\langle i, d, j \rangle$ is an ***angle*** of a node pair $\langle n, n' \rangle$ if there is such a tree X that the following conditions are met.

1. Both n and n' belong to X.
2. Let n^* denote the one of the common ancestors of n and n' in X that has the greatest altitude. Then i [respectively, j] is the difference of the altitudes of n [respectively, n'] and n^*.
3. One of the following, mutually exclusive subconditions is met.
   a. $d = -1$, and n' is a left-relative of n.
   b. $d = 0$, and n' is an ancestor or a descendant of n.
   c. $d = 1$, and n' is a right-relative of n.

Proposition B.21. An integer triple $\langle i, d, j \rangle$ is an angle of a node pair $\langle n, n' \rangle$ if and only if $\langle j, -d, i \rangle$ is an angle of $\langle n', n \rangle$.

Proposition B.22. Each pair of nodes has at most one angle. For any given tree X, it is the case that each pair of the nodes of X has exactly one angle.

Notation B.23. Let n_1 and n_2 be two nodes, and suppose that some tree contains both n_1 and n_2. Then the unique angle of $\langle n_1, n_2 \rangle$ may be denoted as $\sphericalangle(n_1, n_2)$.

Definition B.24. Let $k \in \mathbb{N}$. A given node n_1 is ***k*-close** to a given node n_2 if the pair $\langle n_1, n_2 \rangle$ has such an angle $\langle i, d, j \rangle$ that both $i \leq k$ and $j \leq k$.

Proposition B.25. If a node n_1 is k-close to node n_2 for some $k \geq 0$, then n_1 is k'-close to n_2 for any $k' \geq k$.

Proposition B.26. The 'is k-close to' relation is reflexive and symmetric for any $k \geq 0$, but it is not transitive for any $k > 0$.

B.5 Tree orthoextensions

► Intuitively, the most natural way of tree growth is that a set of new nodes appear as sons of some up-to-now leaf. We call this *orthoextension*.

Definition B.27. A tree X' is a ***direct orthoextension*** of a tree X if there are such a leaf of X and such a non-empty set N of sons of this leaf that $X' = X \cup N$.

Proposition B.28. Let n be a leaf of a given tree X.

1. For any positive integer k, there is such a direct orthoextension of X in which n has exactly k sons.
2. For any son n' of n, there is such a direct orthoextension of X that contains n'.

Definition B.29. A binary relation 'is an ***orthoextension*** of' on trees is defined as the reflexive-transitive closure of the 'is a direct orthoextension of' relation.

Proposition B.30. A tree X' is an orthoextension of a tree X if and only if $X \subseteq X'$.

Proposition B.31. The 'is an orthoextension of' relation is a partial order.

Proposition B.32. The following two statements are equivalent for any two trees X_1 and X_2.

1. $X_1 \cup X_2$ is a tree.
2. There is such a tree X that meets the following conditions.
   a. Both X_1 and X_2 are orthoextensions of X.
   b. Each leaf of X is also a leaf of at least one of X_1 and X_2.

C TREE BELTS AND BELT-SELECTORS

C.1 Belts

Definition C.1. A node set B is a ***belt*** if there is such a tree X that $B \subseteq X$ and each leaf of X has exactly one ancestor in B.

Proposition C.2. For any tree X, both the frontier of X and the one-node set consisting of the root of X are belts.

► By Proposition C.3, it is not counterintuitive to say that "a node set B is a belt of a tree X" when B is both a subset of X and a belt.

Proposition C.3. Suppose that a tree X includes a belt B. Then each leaf of X has exactly one ancestor in B.

C.2 Belt projection

Proposition C.4. The restriction of the 'is a left-relative of' relation to any belt is a total strict-order.

Definition C.5. Any subset of a belt is a ***subbelt***.

Definition C.6. A word w is a ***projection*** of a subbelt N if there are such $k \geq 0$ and such k nodes $n_0, \ldots, n_{k-1}$ that the following conditions are met.

1. $N = \{n_0, \ldots, n_{k-1}\}$.
2. n_{i-1} is a left-relative of n_i for every such i that $0 < i < k$.
3. $w = \alpha(n_0) \cdots \alpha(n_{k-1})$.

Proposition C.7. Any subbelt has exactly one projection.

Notation C.8. The projection of a given subbelt N is denoted as $\pi(N)$.

C.3 Belt wrapping

Definition C.9. A belt B' ***wraps*** a belt B if each node of B' has an ancestor in B.

Proposition C.10. A belt B' wraps a belt B if and only if each node of B has a descendant in B'.

Proposition C.11. The 'wraps' relation on belts is a partial order.

Notation C.12. We may write $B' \succeq B$ or, equivalently, $B \preceq B'$ to state that a given belt B' wraps a given belt B.

Proposition C.13. Let the root of a given tree X be denoted as n_0, and let the frontier of X be denoted as B^*. Then $B^* \succeq B \succeq \{n_0\}$ for any belt B of X.

C.4 Belt-selectors

► Belt-selectors are introduced here, and the examination of their properties will continue in Section E.

Definition C.14. A ***belt-provider*** is any such function $s : \mathcal{X} \times \mathcal{N} \to 2^{\mathcal{N}}$ that meets the following conditions for every tree X and for every node n.

1. If $n \notin X$, then $s(X, n) = \emptyset$.
2. If $n \in X$, then $s(X, n)$ is such a belt of X that contains n.

Proposition C.15. There exists a belt-provider.

Definition C.16. A belt-provider s is ***uniangular***, and hence called a ***belt-selector***, if it meets the following condition.

- Let X_1 and X_2 be two trees. Moreover, let n_1 be a node of X_1, let n_1' be a leaf of X_1, and let the unique ancestor of n_1' in $s(X_1, n_1)$ be denoted as n_1''. Similarly, let n_2 be a node of X_2, let n_2' be a leaf of X_2, and let the unique ancestor of n_2' in $s(X_2, n_2)$ be denoted as n_2''. Then $\lhd(n_1, n_1') = \lhd(n_2, n_2')$ implies $\lhd(n_1, n_1'') = \lhd(n_2, n_2'')$.

Proposition C.17. There exists a belt-selector.

Theorem C.18. Let n_1 and n_1' be nodes in a given tree X_1, and let n_2 and n_2' similarly be nodes in a given tree X_2. Then for any belt-selector s, we have $n_1' \in s(X_1, n_1) \Leftrightarrow n_2' \in s(X_2, n_2)$ if the following conditions are met.

1. $\lhd(n_1, n_1') = \lhd(n_2, n_2')$.
2. n_1' is a leaf of X_1 if and only if n_2' is a leaf of X_2.

Theorem C.19. Let s be a belt-selector, and let n be a node in a given tree X. Then $s(X^*, n) \succeq s(X, n)$ for any orthoextension X^* of X.

C.5 Wings of a belt-selector

Definition C.20. The ***left-wing*** [respectively, ***right-wing***] of a given belt-selector s, denoted as $\overleftarrow{s}$ [respectively, $\overrightarrow{s}$], is the function from $\mathcal{X} \times \mathcal{N}$ to $2^{\mathcal{N}}$ defined as follows: for every tree X and for every node n, the set $\overleftarrow{s}(X, n)$ [respectively, $\overrightarrow{s}(X, n)$] consists of all such nodes in $s(X, n)$ that are left-relatives [respectively, right-relatives] of n.

Proposition C.21. These statements hold for any belt-selector s, for any tree X, and for any node n in X.

1. $s(X, n) = \overleftarrow{s}(X, n) \cup \{n\} \cup \overrightarrow{s}(X, n)$.
2. $\pi(s(X, n)) = \pi(\overleftarrow{s}(X, n))\alpha(n)\pi(\overrightarrow{s}(X, n))$.

Proposition C.22. For any two belt-selectors s_1 and s_2, we have $s_1 = s_2$ if and only if both $\overleftarrow{s_1} = \overleftarrow{s_2}$ and $\overrightarrow{s_1} = \overrightarrow{s_2}$.

D MONOSYSTEMS

► Monosystems and their operation are defined in Sections D.1–D.5. The final Sections D.6 and D.7 discuss two basic properties of monosystems.

D.1 Letter-refiners

Definition D.1. A ***letter-refiner*** is any function from $\mathcal{W} \times \mathcal{U} \times \mathcal{W}$ to $\mathcal{W}$.

Definition D.2. A letter-refiner r is ***isolative*** [respectively, ***left-isolative***, ***right-isolative***] if for every letter c and for every words w_1 and w_2, it is the case that $r(w_1, c, w_2)$ equals $r(\Lambda, c, \Lambda)$ [respectively, $r(\Lambda, c, w_2)$, $r(w_1, c, \Lambda)$].

► So an isolative letter-refiner is simply "context-free".

Definition D.3. A letter-refiner r is ***vigorous*** if $r(w_1, c, w_2) \neq \Lambda$ for every letter c and for every words w_1 and w_2.

Definition D.4. A letter-refiner r ***freezes*** a letter c if $r(w_1, c, w_2) = c$ for every words w_1 and w_2.

Definition D.5. A letter-refiner r ***disregards*** a letter c if the following conditions are met for every words w_1 and w_2.

1. r freezes c.
2. c does not occur in $r(w_1, c_0, w_2)$ for any such letter c_0 that $c_0 \neq c$.
3. $r(wcw_1, c_0, w_2) = r(ww_1, c_0, w_2)$ and $r(w_1, c_0, w_2cw) = r(w_1, c_0, w_2w)$ for every letter c_0 and for every word w.

► So a disregarded letter is frozen, never produced from other letters, and always ignored in the refinement context.

D.2 Constituents of a monosystem

Definition D.6. A ***monosystem*** is any such quintuple $\langle A, M, c_S, r, s\rangle$ that meets the following conditions.

1. The ***alphabet*** A is such a non-empty letter set that $\neg A$ is countably infinite.
2. M is a non-empty subset of A; the members of M and $A \setminus M$ are called ***mutables*** and ***immutables***, respectively.
3. The ***seed-letter*** c_S is a member of M.
4. r is such a letter-refiner that meets the following subconditions.
   a. r disregards every letter in $\neg A$.
   b. r freezes every letter in $A \setminus M$.
   c. r is vigorous.
5. s is a belt-selector.

► The monosystem alphabet need not be finite. Because the complement of the alphabet is still required to be infinite, there are always new, hitherto unused letters that may be added to the alphabet (as confirmed by Propo-

sition D.7 below). Such extensibility of course corresponds well with the intended practical applications of the monosystem model but is admittedly not crucial for the theoretical considerations included in the present report.

► We intentionally deviate from the standard terminology of formal language theory: our mutables and immutables correspond to "nonterminals" and "terminals", respectively, and the seed-letter acts as the "start symbol".

► The letter-refiner of a monosystem must freeze every immutable, but notice that it may well freeze even mutables.

► The letter-refiner of a monosystem must be vigorous, because there is no way to represent the empty word in an expansion tree: every expansion tree node possesses exactly one letter.

Proposition D.7. For any given monosystem $\langle A, M, c_{\mathrm{S}}, r, s\rangle$, there is such a letter set C that meets the following conditions.
1. C is countably infinite.
2. $A \cap C = \emptyset$.
3. $\langle A \cup C, M, c_{\mathrm{S}}, r, s\rangle$ is a monosystem.

D.3 Belt-selector stagnancy

Definition D.8. Let C be a letter set. A given belt-selector s is ***C*-stagnant** at a node n in a tree X if $n \in X$ and if for every orthoextension X^* of X, it is the case that $s(X^*, n) = s(X, n)$ whenever the following condition is met.
- Each such leaf of X whose letter belongs to $\neg C$ is still a leaf of X^*.

► Typically, the letter set referred to in Definition D.8 as 'C' is chosen to be the mutable set of the monosystem under consideration.

Definition D.9. Let C be a letter set. A given tree X is ***C*-compliant** if each such node of X whose letter belongs to $\neg C$ is a leaf of X.

Proposition D.10. Let C be a letter set, and let X and X^* be such two C-compliant trees that X^* is an orthoextension of X. Now if a given belt-selector s is C-stagnant at a given node n in X, then s is C-stagnant at n in X^* and $s(X, n) = s(X^*, n)$.

D.4 Tree expansion by leaf unfolding

Definition D.11. A node n is ***fertile*** in a tree X on a monosystem G if $n \in X$ and the following conditions are met when the mutable set of G is denoted as M.
1. $\alpha(n) \in M$.
2. The belt-selector of G is M-stagnant at n in X.

► For simplicity, Definition D.11 allows even non-leaf nodes to be fertile. Still, the possible fertileness has any significance only when the node is a

leaf.

Definition D.12. A tree X' is ***directly derivable*** from a tree X on a monosystem G if there are such a node n and such a subbelt N that the following conditions are met.

1. n is such a leaf of X that is fertile in X on G.
2. $X' = X \cup N$, and each node in N is a son of n.
3. $\pi(N) = r(\pi(\overleftarrow{s}(X,n)), \alpha(n), \pi(\overrightarrow{s}(X,n)))$ when r and s denote the letter-refiner and the belt-selector of G, respectively.

Definition D.13. For each monosystem G, a binary relation 'is ***derivable*** from, on G' is defined on trees as the reflexive-transitive closure of the 'is directly derivable from, on G' relation.

Proposition D.14. If a tree X' is directly derivable [respectively, is derivable] from a tree X on some monosystem, then X' is a direct orthoextension [respectively, is an orthoextension] of X.

Proposition D.15. For each monosystem G, the 'is derivable from, on G' relation is a partial order.

Definition D.16. A tree X is a ***derivative*** of a monosystem G if X is on G derivable from the one-node tree $\{\langle [\,], c_S \rangle\}$ when c_S denotes the seed-letter of G.

Proposition D.17. Each monosystem has exactly one such derivative that consists of exactly one node, and the only node is fertile in that derivative on the monosystem.

Proposition D.18. Each derivative of a given monosystem is M-compliant when M denotes the mutable set of the monosystem.

D.5 Output extraction

Definition D.19. A word w is an ***export-word*** of a monosystem G if the following conditions are met.

1. Some derivative of G has such a belt whose projection is w.
2. Each letter occurring in w is an immutable of G.

► An export-word is a word "generated" by the monosystem. By condition (2) of Definition D.19, the belt of condition (1) must be the frontier.

Proposition D.20. Λ is not an export-word of any monosystem.

D.6 On confluence

Theorem D.21. Let G be a monosystem, and let the belt-selector of G be denoted as s. Moreover, let X be a derivative of G, let n be a node of X, and let X' be a tree derivable from X on G. Now if n is fertile in X on G, then n

is fertile in X' on G and $s(X,n) = s(X',n)$.

Theorem D.22. Let X_1 and X_2 be two derivatives of a given monosystem G. Then $X_1 \cup X_2$ is such a tree that is derivable both from X_1 and from X_2 on G.

► The property guaranteed by Theorem D.22 might be called monosystem confluence.

Proposition D.23. Each monosystem has at most one export-word.

D.7 On the infertileness of the immutables

► Definition D.11 above implies that immutable-lettered leaves are never fertile. Here we argue that the export-word set would not change even if such leaves were always fertile.

Definition D.24. Let C be a letter set. A given tree X is ***C*-semicompliant** if each such node n of X whose letter belongs to $\neg C$ meets one of the following, mutually exclusive conditions.

1. n is a leaf of X.
2. n has exactly one son in X, and this son possesses the same letter as n.

Proposition D.25. If a tree is C-compliant for a letter set C, then the tree is also C-semicompliant.

Theorem D.26. Let C be a letter set, and let X and X^* be such two C-semicompliant trees that X^* is an orthoextension of X. Moreover, suppose that each such a leaf of X whose letter belongs to C is still leaf of X^*.

1. Let s be a belt-selector, and let n be a node in X.
   a. s is C-stagnant at n in X if and only if s is C-stagnant at n in X^*.
   b. $\pi(\overleftarrow{s}(X,n)) = \pi(\overleftarrow{s}(X^*,n))$ and $\pi(\overrightarrow{s}(X,n)) = \pi(\overrightarrow{s}(X^*,n))$.
2. A given word w is the projection of some belt of X if and only if w is the projection of some belt of X^*.

► For a moment, suppose that immutable-lettered leaves were always fertile. This means that for the mutable set M and for any derivative X, any such M-semicompliant orthoextension X^* of X that still has the same mutable-lettered leaves as X would also be a derivative. Consider then what happens to a given mutable-lettered leaf n of X in the larger X^*. Parts (1a) and (1b) of Theorem D.26 guarantee that neither the fertileness nor the potential unfolding result, respectively, of n would be affected. Therefore, part (2) of the theorem is sufficient to guarantee that the export-word set of the monosystem would not change.

E MORE ON BELT-SELECTORS

► In this Section E, contrary to the later Sections F–I, we consider belt-selectors as independent entities and not as constituents of monosystems.

E.1 Links

Definition E.1. An integer triple is a ***link*** if it is the angle of some node pair.

Proposition E.2. An integer triple $\langle i, d, j \rangle$ is a link if and only if it meets the following conditions.
1. $i \geq 0$, $d \in \{-1, 0, 1\}$, and $j \geq 0$.
2. $d = 0 \Leftrightarrow i \cdot j = 0$.

Definition E.3. A link $\langle i, d, j \rangle$ is ***bipartite*** if $d \neq 0$, and it is ***isosceles*** if $i = j$.

Definition E.4. A given link v' is an ***antilink*** of a given link $v = \langle i, d, j \rangle$ if $v' = \langle j, -d, i \rangle$.

Proposition E.5. Each link has exactly one antilink.

Notation E.6. The antilink of a given link v is denoted as $\neg v$.

Proposition E.7. $\neg(\neg v) = v$ for any link v.

Proposition E.8. For any two nodes n and n' in any single tree, we have $\lhd(n, n') = \neg\lhd(n', n)$.

E.2 Characteristic comb of a belt-selector

Specification E.9. There is such a set that is denoted as ∞ but whose members are left unspecified. (Moreover, it is assumed that $\infty \notin \mathbb{Z}$.)
1. The binary 'less-than' relation is extended from $\mathbb{Z}$ to $\mathbb{Z} \cup \{\infty\}$ by stating that $t < \infty$ for every $t \in \mathbb{Z}$ and by requiring that the relation remains as a total strict-order.
2. The addition operation is extended from $\mathbb{Z}$ to $\mathbb{Z} \cup \{\infty\}$ by stating that $t + \infty = \infty + t = \infty$ for every $t \in \mathbb{Z} \cup \{\infty\}$.

Definition E.10. A ***comb*** is any function from $\mathbb{N}^+ \times \{-1, 1\}$ to $\mathbb{N}^+ \cup \{\infty\}$.

Definition E.11. A comb f is a ***characteristic comb*** of a belt-provider s if for every tree X, for every node n of X, and for every leaf n' of X, the following conditions are met when $\lhd(n, n')$ is denoted as $\langle i, d, j \rangle$ and the unique ancestor of n' that belongs to $s(X, n)$ is denoted as n''.
1. Suppose $d \neq 0$ and $j \leq f(i, d)$. Then $n'' = n'$.
2. Suppose $d \neq 0$ and $j > f(i, d)$. Then n'' is the unique proper ancestor of n' for which $\lhd(n, n'') = \langle i, d, f(i, d) \rangle$.

Notation E.12. The set of belt-providers [respectively, belt-selectors, combs] is denoted as $\mathcal{P}$ [respectively, $\mathcal{S}$, $\mathcal{F}$].

Theorem E.13. Let R denote the set of all such members $\langle f, s\rangle$ of $\mathcal{F} \times \mathcal{P}$ that f is a characteristic comb of s. Then $R \subseteq \mathcal{F} \times \mathcal{S}$, and moreover, R is a bijective function from $\mathcal{F}$ to $\mathcal{S}$.

Notation E.14. The unique characteristic comb of a given belt-selector s is denoted as ϕ_s.

Proposition E.15. Let s_1 and s_2 be two belt-selectors.

1. $\overleftarrow{s_1} = \overleftarrow{s_2}$ if and only if $\phi_{s_1}(i, -1) = \phi_{s_2}(i, -1)$ for every $i > 0$.
2. $\overrightarrow{s_1} = \overrightarrow{s_2}$ if and only if $\phi_{s_1}(i, 1) = \phi_{s_2}(i, 1)$ for every $i > 0$.

E.3 Locks and hooks of a belt-selector

Definition E.16. Let s be a belt-selector, and let $v = \langle i, d, j\rangle$ be a link.

1. v is a ***lock*** of s if v is bipartite and $\phi_s(i, d) = j$.
2. v is a ***hook*** of s if v is bipartite and $\phi_s(i, d) > j$.

Notation E.17. The lock set [respectively, hook set] of a given belt-selector s is denoted as $\lambda(s)$ [respectively, $\chi(s)$].

Proposition E.18. $\lambda(s) \cap \chi(s) = \emptyset$ for any belt-selector s.

Proposition E.19. Let s_1 and s_2 be two belt-selectors.

1. $s_1 = s_2$ if and only if $\phi_{s_1} = \phi_{s_2}$.
2. $s_1 = s_2$ if and only if $\lambda(s_1) = \lambda(s_2)$.
3. $s_1 = s_2$ if and only if $\chi(s_1) = \chi(s_2)$.

Proposition E.20. Let s be a belt-selector, and let n and n' be nodes in a given tree X. Then $n' \in s(X, n)$ if and only if one of the following, mutually exclusive conditions is met.

1. $n' = n$.
2. $\lhd(n, n') \in \lambda(s)$.
3. $\lhd(n, n') \in \chi(s)$, and n' is a leaf of X.

► Proposition E.21 paraphrases the original Definition D.8 in a more succinct form.

Proposition E.21. Let C be a letter set. Then a belt-selector s is C-stagnant at a node n in a tree X if and only if $n \in X$ and each node $n' \in s(X, n) \setminus \{n\}$ meets at least one of the following conditions.

1. $\alpha(n') \in \neg C$.
2. $\lhd(n, n') \in \lambda(s)$.

E.4 Operations on belt-selectors

E.4.1 Junction of two belt-selectors

Definition E.22. The ***junction*** of a given belt-selector s_1 with a given belt-selector s_2 is denoted as $s_1 \| s_2$ and defined as the unique belt-selector that meets the following conditions for every $i > 0$.

1. $\phi_{s_1 \| s_2}(i, -1) = \phi_{s_1}(i, -1)$.
2. $\phi_{s_1 \| s_2}(i, 1) = \phi_{s_2}(i, 1)$.

Proposition E.23. Let s_1, s_2, and s_3 be belt-selectors.

1. $s_1 \| s_1 = s_1$.
2. $\overleftarrow{s_1 \| s_2} = \overleftarrow{s_1}$ and $\overrightarrow{s_1 \| s_2} = \overrightarrow{s_2}$.
3. $(s_1 \| s_2) \| s_3 = s_1 \| (s_2 \| s_3) = s_1 \| s_3$.

E.4.2 Mirror-image

Definition E.24. The ***mirror-image*** of a given belt-selector s is denoted as ${}^{\perp}s$ and defined as the unique such belt-selector that $\phi_{{}^{\perp}s}(i, d) = \phi_s(i, -d)$ for every $\langle i, d \rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

Proposition E.25. ${}^{\perp}({}^{\perp}s) = s$ for any belt-selector s.

Definition E.26. A belt-selector s is ***symmetric*** if ${}^{\perp}s = s$.

Proposition E.27. Let s_1 and s_2 be belt-selectors.

1. ${}^{\perp}(s_1 \| s_2) = ({}^{\perp}s_2) \| ({}^{\perp}s_1)$.
2. If both s_1 and s_2 are symmetric, then ${}^{\perp}(s_1 \| s_2) = s_2 \| s_1$.
3. $s_1 \| ({}^{\perp}s_1)$ is symmetric.

E.4.3 Incrementation

Definition E.28. The ***incrementation*** of a given belt-selector s is denoted as $\triangle s$ and defined as the unique such belt-selector that $\phi_{\triangle s}(i, d) = \phi_s(i, d) + 1$ for every $\langle i, d \rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

Proposition E.29. A belt-selector s is symmetric if and only if $\triangle s$ is symmetric.

E.4.4 Convexification

Definition E.30. A belt-selector s is ***convex*** if $\phi_s(i + 1, d) \geq \phi_s(i, d)$ for every $\langle i, d \rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

Definition E.31. The ***convexification*** of a given belt-selector s is denoted as $\odot s$ and defined as the unique belt-selector that meets the following conditions for every $\langle i, d \rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

1. If $i = 1$, then $\phi_{\odot s}(i, d) = \phi_s(i, d)$.

2. If $i > 1$, then $\phi_{\odot s}(i, d)$ is the maximum of $\phi_s(i, d)$ and $\phi_{\odot s}(i - 1, d)$.

Proposition E.32. Let s and s' be belt-selectors.

1. $\odot s$ is convex.
2. s is convex if and only if $\odot s = s$.
3. s is convex if and only if ${}^{\perp}s$ is convex.
4. s is convex if and only if $\triangle s$ is convex.
5. If both s and s' are convex, then $s \| s'$ is convex.

E.5 Belt-selector wrapping

Definition E.33. A belt-selector s' ***wraps*** a belt-selector s if for every node n of every tree X, we have $s'(X, n) \succeq s(X, n)$.

► So we say that one belt-selector wraps another if the belt selected by the former always wraps the belt selected by the latter.

Proposition E.34. Let s and s' be two belt-selectors.

1. s' wraps s if and only if $\phi_{s'}(i, d) \geq \phi_s(i, d)$ for every $\langle i, d \rangle \in \mathbb{N}^+ \times \{-1, 1\}$.
2. s' wraps s if and only if $\chi(s) \subseteq \chi(s')$.

Proposition E.35. The 'wraps' relation on belt-selectors is a partial order.

Notation E.36. We may write $s' \succeq s$ or, equivalently, $s \preceq s'$ to state that a given belt-selector s' wraps a given belt-selector s.

Proposition E.37. Let s_1 and s_2 be two belt-selectors. Then there are unique such belt-selectors s^* and s^{**} that $\chi(s^*) = \chi(s_1) \cup \chi(s_2)$ and $\chi(s^{**}) = \chi(s_1) \cap \chi(s_2)$. Moreover, these s^* and s^{**} are the unique belt-selectors that meet the following conditions.

1. $s^* \succeq s_1 \succeq s^{**}$ and $s^* \succeq s_2 \succeq s^{**}$.
2. The following subconditions are met for every belt-selector s'.
   a. If $s' \succeq s_1$ and $s' \succeq s_2$, then $s' \succeq s^*$.
   b. If $s_1 \succeq s'$ and $s_2 \succeq s'$, then $s^{**} \succeq s'$.

Proposition E.38. Let s, s', s_0, and s'_0 be belt-selectors.

1. $\triangle s \succeq s$.
2. $\odot s \succeq s$.
3. $s \succeq s'$ if and only if ${}^{\perp}s \succeq {}^{\perp}s'$.
4. $s \succeq s'$ if and only if $\triangle s \succeq \triangle s'$.
5. If $s \succeq s'$, then $\odot s \succeq \odot s'$.
6. If $s \succeq s'$ and $s_0 \succeq s'_0$, then $s \| s_0 \succeq s' \| s'_0$.

Proposition E.39. Let s and s^* be such two belt-selectors that $s^* \succeq s$. Now if s^* is C-stagnant at a given node n in a given tree X for a given letter set C, then s is also C-stagnant at n in X.

E.6 Some individual belt-selectors

Definition E.40. Let $k \in \mathbb{N}^+$. The ***extroversive*** [respectively, ***centroversive***, k***-tubuloversive***, ***introversive***] belt-selector is denoted as σ_{E} [respectively, σ_{C}, $\sigma_{\mathsf{T}(k)}$, σ_{I}] and defined uniquely by the appropriate one of the following requirements, which concern every $\langle i, d\rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

1. Extroversive: $\phi_{\sigma_{\mathsf{E}}}(i, d) = \infty$.
2. Centroversive: $\phi_{\sigma_{\mathsf{C}}}(i, d) = i$.
3. k-tubuloversive: $\phi_{\sigma_{\mathsf{T}(k)}}(i, d)$ is the minimum of i and k.
4. Introversive: $\phi_{\sigma_{\mathsf{I}}}(i, d) = 1$.

Definition E.41. A belt-selector is ***tubuloversive*** if it is k-tubuloversive for some $k \in \mathbb{N}^+$. The set of all tubuloversive belt-selectors is denoted as Σ_{T}.

Proposition E.42. $\sigma_{\mathsf{I}} = \sigma_{\mathsf{T}(1)} \in \Sigma_{\mathsf{T}}$.

Proposition E.43. Any member of $\{\sigma_{\mathsf{E}}, \sigma_{\mathsf{C}}\} \cup \Sigma_{\mathsf{T}}$ is both symmetric and convex.

Proposition E.44. The following statements hold for any $k \geq 1$.

1. $\sigma_{\mathsf{E}} \succeq \sigma_{\mathsf{C}} \succeq \sigma_{\mathsf{T}(k+1)} \succeq \sigma_{\mathsf{T}(k)}$.
2. $\sigma_{\mathsf{E}} \neq \sigma_{\mathsf{C}} \neq \sigma_{\mathsf{T}(k+1)} \neq \sigma_{\mathsf{T}(k)}$.

Proposition E.45. $\sigma_{\mathsf{E}} \succeq s \succeq \sigma_{\mathsf{I}}$ for any belt-selector s.

F PROGRESSIVENESS

F.1 Weak progressiveness

Definition F.1. A belt-selector s is ***weakly progressive*** if every derivative X of every such monosystem G whose belt-selector is s meets the following condition.

- If X has such a leaf that possesses a mutable of G, then some leaf of X is fertile in X on G.

Definition F.2. A finite sequence z of links is a ***circuit*** if it meets the following conditions.

1. $|z| \geq 2$.
2. Each link occurring in z is bipartite.
3. There are such a tree X and such $|z|+1$ leaves $n_0, \ldots, n_{|z|}$ of X that the following subconditions are met.
   a. $n_0 = n_{|z|}$.
   b. $\lhd(n_i, n_{i+1}) = z(i)$ for every $0 \leq i < |z|$.

Proposition F.3. If z is such a circuit that $|z| = 2$, then $z(0) = \neg z(1)$.

Definition F.4. A circuit z is ***rudimentary*** if $|z| = 2$ and the two links $z(0)$ and $z(1)$ are both isosceles.

Definition F.5. A circuit is a ***trap*** of a given belt-selector if each link occurring in the circuit is a hook of the belt-selector.

Proposition F.6. Let z be such a finite link sequence that $|z| \geq 2$, let i be such an integer that $0 < i < |z|$, and let z_i denote the finite link sequence $[\, z(i), \ldots, z(|z|-1) \,] \oplus [\, z(0), \ldots, z(i-1) \,]$.

1. z is a circuit if and only if z_i is a circuit.
2. z is a trap of a given belt-selector if and only if z_i is a trap of the same belt-selector.

Proposition F.7. A belt-selector s has a rudimentary trap if and only if there is such $i > 0$ that both $\phi_s(i, -1) > i$ and $\phi_s(i, 1) > i$.

Theorem F.8. For each $k \geq 2$, there is such a convex belt-selector s_k that has a trap whose length is k but no such trap whose length is smaller than k.

Theorem F.9. A belt-selector s has a trap if and only if at least one of the following conditions is met.

1. s has a rudimentary trap.
2. There is such a hook $\langle i, d, j \rangle$ of s that $i < j$ and that $\langle i', -d, i \rangle$ is a hook of s for some $i' > 0$.

Theorem F.10. The following three statements are equivalent for any belt-selector s.

1. s is weakly progressive.
2. s has no traps.
3. The following conditions are met for every $\langle i, d \rangle \in \mathbb{N}^+ \times \{-1, 1\}$.

a. If $\phi_s(i,d) > i$, then $\phi_s(i,-d) \leq i$.
b. If $\phi_s(i,d) > i+1$, then $\phi_s(i',-d) \leq i$ for every $i' > 0$.

F.2 Strong progressiveness

Definition F.11. A belt-selector s is ***strongly progressive*** if every derivative X of every such monosystem G whose belt-selector is s meets the following condition.

- If X has such a leaf n that possesses a mutable of G, then there is such a tree X^* derivable from X on G that n is fertile in X^* on G.

Proposition F.12. Every strongly progressive belt-selector is weakly progressive.

Theorem F.13. A belt-selector s is strongly progressive if and only if s has no rudimentary trap and $\triangle\sigma_{\mathsf{C}} \succeq s$.

F.3 Distributive progressiveness

Theorem F.14. σ_{I} is the only such belt-selector s that every derivative X of every such monosystem G whose belt-selector is s meets the following condition.

- Every such leaf of X that possesses a mutable of G is fertile in X on G.

Definition F.15. Let $k \in \mathbb{N}$. A given belt-selector s is ***k-distributively progressive*** if every derivative X of every such monosystem G whose belt-selector is s meets the following condition.

- For every such leaf n of X that possesses a mutable of G, there is such a leaf of X that is both fertile in X on G and k-close to n.

Proposition F.16. If a belt-selector is k-distributively progressive for some $k \geq 0$, then it is k'-distributively progressive for any $k' \geq k$.

Proposition F.17. σ_{I} is the only 0-distributive belt-selector.

Definition F.18. A belt-selector is said to be ***distributively progressive*** if it is k-distributively progressive for some $k \in \mathbb{N}$.

► By comparing Definitions F.15 and F.1, one easily sees that distributive progressiveness implies weak progressiveness. The following Proposition F.19 is a stronger claim.

Proposition F.19. Every distributively progressive belt-selector is strongly progressive.

Theorem F.20. Let $k \geq 0$. Then a strongly progressive belt-selector s is k-distributively progressive if and only if it meets the following condition.

- Suppose that s has a hook $\langle i,d,j\rangle$. Then s also has such a hook $\langle i^*,d,j\rangle$ that $i^* \leq k$.

► The following Proposition F.21 guarantees that the countably infinite hierarchy within the class of distributively progressive belt-selectors does not collapse.

Proposition F.21. The following statements hold.

1. $\sigma_{\mathsf{T}(1)}$ is 0-distributively progressive.
2. $(\triangle\sigma_{\mathsf{T}(1)})\|\sigma_{\mathsf{T}(1)}$ is 1-distributively progressive but not 0-distributively progressive.
3. Let $k \geq 2$. Then both $\sigma_{\mathsf{T}(k)}$ and $(\triangle\sigma_{\mathsf{T}(k)})\|\sigma_{\mathsf{T}(k)}$ are k-distributively progressive, but neither one is $(k-1)$-distributively progressive.

Theorem F.22. A belt-selector s is distributively progressive if and only if s has no rudimentary trap and $\triangle\sigma_{\mathsf{T}(k)} \succeq s$ for some $k \geq 1$.

F.4 Rigidness

Definition F.23. A belt-selector s is ***rigid*** if every derivative X of every such monosystem G whose belt-selector is s has at most one such leaf that is fertile in X on G.

Proposition F.24. No distributively progressive belt-selector is rigid.

Theorem F.25. A belt-selector s is rigid if and only if every bipartite link $\langle i, d, j\rangle$ meets at least one of the following conditions.

1. At least one of $\langle i, d, j\rangle$ and its antilink is a hook of $\odot s$.
2. There are such positive integers $i' < i$ and $j' < j$ that both $\langle i', d, j'\rangle$ and its antilink are hooks of s.

F.5 Further results

Proposition F.26. Let s and s' be such two belt-selectors that $s \succeq s'$. Now if s is weakly progressive [respectively, strongly progressive, distributively progressive, not rigid], then even s' is weakly progressive [respectively, strongly progressive, distributively progressive, not rigid].

Proposition F.27. Let $G = \langle A, M, c_{\mathrm{S}}, r, s\rangle$ be a given monosystem, and let n be a node in a given derivative X of G. Then n is fertile in X on G if and only if n is fertile in X on the monosystem $\langle A, M, c_{\mathrm{S}}, r, \odot s\rangle$.

► Let us briefly recall trisystems, and in particular, equivalence of their frames, which was discussed in Section 6.2. It follows rather directly from Proposition F.27 that two trisystem frames are strongly equivalent if one is obtained from the other by replacing the threshold-selector with its convexification.

Proposition F.28. Let s be a belt-selector. Now if one of s, ${}^{\perp}s$, and $\odot s$ is weakly progressive [respectively, strongly progressive, distributively progressive, rigid], then so are even the other two.

Proposition F.29. These statements hold.

1. σ_{E} is rigid but not weakly progressive.
2. $\sigma_{\mathsf{E}}||\sigma_{\mathsf{I}}$ is weakly progressive and rigid but not strongly progressive.
3. $(\triangle\sigma_{\mathsf{C}})||\sigma_{\mathsf{C}}$ is strongly progressive and rigid.
4. σ_{C} is strongly progressive but neither distributively progressive nor rigid.
5. Both $\sigma_{\mathsf{T}(k)}$ and $(\triangle\sigma_{\mathsf{T}(k)})||\sigma_{\mathsf{T}(k+1)}$ are distributively progressive for every $k \geq 1$.

► By Proposition F.29, the four-level hierarchy constituted by unrestricted, weakly progressive, strongly progressive, and distributively progressive belt-selectors does not collapse.

F.6 Idealness

Definition F.30. A belt-selector is ***ideal*** if it is convex, weakly progressive, and rigid.

Proposition F.31. A belt-selector s is ideal if and only if ${}^{\perp}s$ is ideal.

Proposition F.32. Both $\sigma_{\mathsf{E}}||\sigma_{\mathsf{I}}$ and $(\triangle\sigma_{\mathsf{C}})||\sigma_{\mathsf{C}}$ are ideal.

Theorem F.33. A belt-selector s is ideal if and only if the following conditions are met.

1. For every bipartite link v, exactly one of v and $\neg v$ is a hook of s.
2. s has no such lock $\langle i, d, j\rangle$ that $j > i + 1$.

Proposition F.34. No ideal belt-selector is symmetric.

Theorem F.35. A belt-selector s is ideal if and only if s is weakly progressive and the following further condition is met for every such monosystem G whose belt-selector is s, for every derivative X of G, and for every leaf n of X.

- If n is fertile in X on G, then $s(X, n)$ is the frontier of X.

Theorem F.36. A belt-selector s is ideal if and only if there are such $t^* \in \mathbb{N}^+ \cup \{\infty\}$, such $d^* \in \{-1, 1\}$, and such a function $b^* : \mathbb{N}^+ \to \{-1, 1\}$ that the following conditions are met for every $i > 0$.

1. If $i < t^*$, then $\phi_s(i, -b^*(i)) = i$ and $\phi_s(i, b^*(i)) = i + 1$.
2. If $i \geq t^*$, then $\phi_s(i, -d^*) = t^*$ and $\phi_s(i, d^*) = \infty$.

Theorem F.37. For each weakly progressive [respectively, strongly progressive] belt-selector s, there is such an ideal [respectively, ideal and strongly progressive] belt-selector that wraps s.

G STABLENESS

Definition G.1. Let c be a letter. A given monosystem G^* is a ***c*-alteration** of a given monosystem $G = \langle A, M, c_{\mathrm{S}}, r, s\rangle$ if there is such a letter-refiner r^* that the following conditions are met.

1. $G^* = \langle A, M, c_{\mathrm{S}}, r^*, s\rangle$.
2. $r^*(w_1, c', w_2) = r(w_1, c', w_2)$ for every such letter c' that $c' \neq c$ and for every two words w_1 and w_2.

► So a c-alteration differs from the original monosystem in no other way than by perhaps having a different refinement rule for letter c. (In practice, the notion of a c-alteration is thus meaningful only when c is a mutable.)

Definition G.2. Let $k \in \mathbb{N}$. A given belt-selector s is ***k*-stable** if the following condition is met for every such monosystem G whose belt-selector is s, for every derivative X of G, and for every mutable c of G.

- Suppose that X contains exactly one such node n that possesses c. Moreover, let N^* be the set of all the nodes of X that have no such ancestor in X that is k-close to n. Then each c-alteration of G has such a derivative that includes N^*.

Proposition G.3. If a belt-selector is k-stable for some $k \geq 0$, then it is k'-stable for any $k' \geq k$.

Definition G.4. A belt-selector is said to be ***stable*** if it is k-stable for some $k \in \mathbb{N}$.

Theorem G.5. Let s be a weakly progressive belt-selector.

1. s is 0-stable if and only if $s = \sigma_{\mathsf{I}}$.
2. Let $k \geq 1$. Then s is k-stable if and only if $\triangle\sigma_{\mathsf{T}(k)} \trianglerighteq s$.

Theorem G.6. A weakly progressive belt-selector is stable if and only if it is distributively progressive.

H SOUNDNESS

H.1 Basic definitions and results

Definition H.1. A ***semantic classifier***, or simply a ***classifier***, is any equivalence relation on $\mathcal{W}$.

Definition H.2. Let e be a classifier. Two given words w_1 and w_2 are ***e-equivalent***, which is denoted as $w_1 \overset{e}{\sim} w_2$, if $\langle w_1, w_2 \rangle \in e$.

Definition H.3. Let e be a classifier. A given monosystem G is ***e-sound*** if $c_S \overset{e}{\sim} w$ for the seed-letter c_S of G and for each export-word w of G.

► Of course, each monosystem has at most one export-word by Proposition D.23, contrary to what the wording of Definition H.3 might suggest.

Definition H.4. Let e be a classifier. A given letter-refiner r is ***e-sound*** if $w_1 c w_2 \overset{e}{\sim} w_1 r(w_1, c, w_2) w_2$ for every letter c and for every words w_1 and w_2.

Definition H.5. A belt-selector s is ***sound*** if the following condition is met for every such monosystem G whose belt-selector is s and for every classifier e.

- If the letter-refiner of G is e-sound, then G is e-sound.

Theorem H.6. Every sound belt-selector is rigid.

Theorem H.7. A weakly progressive belt-selector is sound if and only if it is ideal.

H.2 Rewriting maps

Definition H.8. A ***rewriting map***, or simply a ***map***, is any subset of $\mathcal{W}^4$. When m_1 and m_2 are such two maps that $m_1 \subseteq m_2$, we say that m_1 is a ***submap*** of m_2, and that m_2 is a ***supermap*** of m_1.

Notation H.9. Suppose that a given word quadruple $\langle w^*, w_1, w, w_2 \rangle$ belongs to a given map m. We denote this fact as $w^* \lhd m\lfloor w_1, w, w_2 \rfloor$.

Definition H.10. Let e be a classifier. A given map m is ***e-sound*** if the following condition is met for every words w, w_1, w_2, and w^*.

- If $w^* \lhd m\lfloor w_1, w, w_2 \rfloor$, then $w_1 w w_2 \overset{e}{\sim} w_1 w^* w_2$.

Proposition H.11. If a map m is e-sound for a classifier e, then every submap of m is e-sound.

Definition H.12. Let m be a map. A given letter-refiner r is ***m-legitimate*** if $r(w_1, c, w_2) \lhd m\lfloor w_1, c, w_2 \rfloor$ for every letter c and for every words w_1 and w_2.

Proposition H.13. If a letter-refiner r is m-legitimate for a map m, then r is m^*-legitimate for every supermap m^* of m.

Proposition H.14. Let r be a letter-refiner, let m be a map, and let e be a classifier. Now if r is m-legitimate and m is e-sound, then r is e-sound.

Definition H.15. Let m be a map. A given belt-selector s is ***m*-wise sound** if the following condition is met for every such monosystem G whose belt-selector is s and for every classifier e.

- If the letter-refiner of G is m-legitimate and if m is e-sound, then G is e-sound.

Proposition H.16. A belt-selector is sound if and only if it is m-wise sound for every map m.

H.3 Reflexive, transitive, and adjunctive maps

Definition H.17. A map m is ***reflexive*** if it meets the following condition for every words w_1, w, and w_2.

- $w \lhd m\lfloor w_1, w, w_2\rfloor$.

Definition H.18. A map m is ***transitive*** if it meets the following condition for every words w_1, w, w_2, w^*, and w^{**}.

- If both $w^* \lhd m\lfloor w_1, w, w_2\rfloor$ and $w^{**} \lhd m\lfloor w_1, w^*, w_2\rfloor$, then even $w^{**} \lhd m\lfloor w_1, w, w_2\rfloor$.

Definition H.19. A map m is ***adjunctive*** if it meets the following condition for every words w_{11}, w_1, w_2, w_{22}, w_1^*, and w_2^*.

- If both $w_1^* \lhd m\lfloor w_{11}, w_1, w_2 w_{22}\rfloor$ and $w_2^* \lhd m\lfloor w_{11} w_1, w_2, w_{22}\rfloor$, then even $w_1^* w_2^* \lhd m\lfloor w_{11}, w_1 w_2, w_{22}\rfloor$.

Theorem H.20. Let $k \geq 1$.

1. σ_{C} is m-wise sound for every such map m that is adjunctive but σ_{I} is not.
2. $\sigma_{\mathsf{T}(k)}$ is m-wise sound for every such map m that is transitive and adjunctive.
3. $(\triangle\sigma_{\mathsf{T}(k)})\|\sigma_{\mathsf{T}(k+1)}$ and $\sigma_{\mathsf{T}(k+1)}\|(\triangle\sigma_{\mathsf{T}(k)})$ are m-wise sound for every such map m that is reflexive, transitive, and adjunctive.

H.4 Civil, semicivil, and semigentle maps

Definition H.21. A map m is ***left-civil*** [respectively, ***right-civil***] if the following condition is met for every words w_{11}, w_1, w, w_2, w_{22}, w_1^*, and w_2^*.

- If both $w_1^* \lhd m\lfloor w_{11}, w_1, w w_2 w_{22}\rfloor$ and $w_2^* \lhd m\lfloor w_{11} w_1 w, w_2, w_{22}\rfloor$, then the appropriate one of the following statements holds.
  a. Left-civilness: $w_1^* \lhd m\lfloor w_{11}, w_1, w w_2^* w_{22}\rfloor$.
  b. Right-civilness: $w_2^* \lhd m\lfloor w_{11} w_1^* w, w_2, w_{22}\rfloor$.

Definition H.22. Let m be a map.

1. m is ***semicivil*** if m is left-civil or right-civil.
2. m is ***civil*** if m is both left-civil and right-civil.

Theorem H.23. Suppose that a belt-selector s is m-wise sound for every

adjunctive map m. Then s is m^*-wise sound for every semicivil map m^*.

Theorem H.24. σ_{C} is m-wise sound for every semicivil map m but σ_{I} is not.

Definition H.25. A map m [respectively, m'] is ***left-gentle*** [respectively, ***right-gentle***] if it has such a supermap m' [respectively, m] that the following condition is met for every words w_{11}, w_1, w, w_2, w_{22}, w_1^*, and w_2^*.

- If both $w_1^* \lhd m\lfloor w_{11}, w_1, ww_2w_{22}\rfloor$ and $w_2^* \lhd m'\lfloor w_{11}w_1w, w_2, w_{22}\rfloor$, then both $w_1^* \lhd m\lfloor w_{11}, w_1, ww_2^*w_{22}\rfloor$ and $w_2^* \lhd m'\lfloor w_{11}w_1^*w, w_2, w_{22}\rfloor$.

Definition H.26. A map is ***semigentle*** if it is left-gentle or right-gentle.

Proposition H.27. Every left-gentle [respectively, right-gentle, semigentle] map is also left-civil [respectively, right-civil, semicivil].

► By Theorem H.28, there is no need to give the conjunction of left-gentleness and right-gentleness any special name (such as "gentleness").

Theorem H.28. A map is civil if and only if it is both left-gentle and right-gentle.

H.5 On smooth belt-selectors

Definition H.29. A belt-selector s is ***smooth*** if the following condition is met for every tree X and for every nodes n and n' in X.

- If n is a leaf of X and if n' has a proper descendant in $s(X, n)$, then $s(X, n) \succeq s(X, n')$.

Proposition H.30. A belt-selector s is smooth if and only if ${}^{\perp}s$ is smooth.

Theorem H.31. These statements hold.

1. Every smooth belt-selector is convex.
2. Every such belt-selector that is convex and strongly progressive is smooth.
3. There is such a convex and weakly progressive belt-selector that is not smooth.
4. Every ideal belt-selector is smooth.

Theorem H.32. Every smooth belt-selector is m-wise sound for every civil map m.

H.6 On subideal belt-selectors

Definition H.33. A belt-selector s is ***subideal*** if there are such two ideal belt-selectors s_1 and s_2 that $s = s_1||s_2 \preceq s_2||s_1$.

Proposition H.34. A belt-selector s is subideal if and only if ${}^{\perp}s$ is subideal.

Proposition H.35. A belt-selector is subideal if and only if it can be expressed as $s_1||s_2$ for such two ideal belt-selectors s_1 and s_2 that $s_1||\sigma_{\mathsf{E}} \preceq$

$s_2 \| \sigma_{\mathsf{E}}$.

Proposition H.36. These statements hold.

1. Every ideal belt-selector is subideal.
2. Every member of $\{\sigma_{\mathsf{C}}\} \cup \Sigma_{\mathsf{T}}$ is subideal.
3. $(\triangle \sigma_{\mathsf{T}(k)}) \| \sigma_{\mathsf{T}(k+1)}$ is subideal for every $k \geq 1$.

Proposition H.37. Every subideal belt-selector is convex and weakly progressive.

Theorem H.38. Every subideal belt-selector is smooth.

Theorem H.39. Every subideal belt-selector is m-wise sound for every semigentle map m.

I COALESCENCE

Definition I.1. Let N be a subbelt, and let n be a node. The ***left-tail*** of the pair $\langle N, n\rangle$ is denoted as $N{\restriction}n$ and defined as the subbelt that consists of every such node that both belongs to N and is a left-relative of n.

Proposition I.2. For any tree X, for any node n in X, and for any belt-selector s, we have $s(X,n){\restriction}n = \overleftarrow{s}\,(X,n)$.

Definition I.3. A belt-selector s is ***left-coalescent*** if the following condition is met for every tree X, for every nodes n_1 and n_2 in X, and for every node n in $\overleftarrow{s}\,(X,n_1) \cap \overleftarrow{s}\,(X,n_2)$.

- Let the two angles $\sphericalangle(n_1,n)$ and $\sphericalangle(n_2,n)$ be denoted as $\langle i_1,-1,j_1\rangle$ and $\langle i_2,-1,j_2\rangle$, respectively. Then $\phi_s(i_1,-1)-j_1 = \phi_s(i_2,-1)-j_2$ implies $\overleftarrow{s}\,(X,n_1){\restriction}n = \overleftarrow{s}\,(X,n_2){\restriction}n$.

Proposition I.4. A belt-selector s is left-coalescent if and only if $s\|s^*$ is left-coalescent for every belt-selector s^*.

Definition I.5. Let s be a belt-selector.

1. s is ***right-coalescent*** if ${}^{\perp}s$ is left-coalescent.
2. s is ***coalescent*** if it is both left-coalescent and right-coalescent.

► By Definition I.5, any result we may achieve on left-coalescence easily carries over to both right-coalescence and coalescence. Hence, defining the left-tail (in Definition I.1 above) but not its right-hand counterpart is indeed sufficient for our present purposes.

Proposition I.6. A belt-selector s is coalescent if and only if ${}^{\perp}s$ is coalescent.

► Of course, the following Theorem I.7 is important because of Theorem D.21.

Theorem I.7. Let G be a monosystem, let X be a derivative of G, and let n_1 and n_2 be two nodes of X. Moreover, suppose that both n_1 and n_2 are fertile in X on G. Finally, let the belt-selector of G be denoted as s. Then for any node n in $\overleftarrow{s}\,(X,n_1) \cap \overleftarrow{s}\,(X,n_2)$, we have $\overleftarrow{s}\,(X,n_1){\restriction}n = \overleftarrow{s}\,(X,n_2){\restriction}n$ if at least one of the following conditions is met.

1. $\overleftarrow{s} \in \{\overleftarrow{\sigma_{\mathsf{E}}}, \overleftarrow{\sigma_{\mathsf{I}}}\}$.
2. s is left-coalescent, and $\alpha(n)$ is a mutable of G.
3. s is left-coalescent, $\alpha(n)$ is an immutable of G, and at least one of the following subconditions is met when $\sphericalangle(n_1,n)$ and $\sphericalangle(n_2,n)$ are denoted as $\langle i_1,-1,j_1\rangle$ and $\langle i_2,-1,j_2\rangle$, respectively.
   a. $\phi_s(i_1,-1) = \phi_s(i_2,-1) = \infty$.
   b. $\phi_s(i_1,-1) - j_1 = \phi_s(i_2,-1) - j_2$.

Theorem I.8. A belt-selector s is left-coalescent if and only if the following conditions are met by every positive integers i_1 and i_2.

1. Suppose $\phi_s(i_1,-1) = \infty$. Then we have $\phi_s(i^*,-1) = \infty$ for every

$i^* \geq i_1$.

2. Suppose $\phi_s(i_1, -1) = \phi_s(i_2, -1)$. Then we have $\phi_s(i_1 + k, -1) = \phi_s(i_2 + k, -1)$ for every $k > 0$.
3. Suppose $\phi_s(i_1, -1) < \phi_s(i_2, -1) < \infty$. Then we have $\phi_s(i^* + 1, -1) = \phi_s(i^*, -1) + 1$ for every such i^* that $i_1 \leq i^* < i_1 + \phi_s(i_2, -1) - \phi_s(i_1, -1)$.

Proposition I.9. Any member of $\{\sigma_{\mathsf{E}}, \sigma_{\mathsf{C}}\} \cup \Sigma_{\mathsf{T}}$ is left-coalescent.

Proposition I.10. A belt-selector s is left-coalescent if and only if $\triangle s$ is left-coalescent.

Theorem I.11. A belt-selector s is both ideal and coalescent if and only if there are such t^* in $\mathbb{N}^+ \cup \{\infty\}$ and such d_1^* and d_2^* in $\{-1, 1\}$ that the following conditions are met for every $i > 0$.

1. If $i < t^*$, then $\phi_s(i, -d_1^*) = i$ and $\phi_s(i, d_1^*) = i + 1$.
2. If $i \geq t^*$, then $\phi_s(i, -d_2^*) = t^*$ and $\phi_s(i, d_2^*) = \infty$.

Theorem I.12. An ideal belt-selector is left-coalescent if and only if it is right-coalescent.

Theorem I.13. A belt-selector s is both subideal and coalescent if and only if there are such belt-selectors s_1 and s_2 that meet the following conditions.

1. $s = s_1 \| s_2 \preceq s_2 \| s_1$.
2. Both s_1 and s_2 are ideal.
3. Both s_1 and s_2 are coalescent.

Proposition I.14. There is such a coalescent belt-selector that is distributively progressive but not convex.

HELSINKI UNIVERSITY OF TECHNOLOGY LABORATORY FOR THEORETICAL COMPUTER SCIENCE
TECHNICAL REPORTS

HUT-TCS-B7 Victor Varshavsky
Circuits Insensitive to Delays in Transistors and Wires. November 1989.

HUT-TCS-B8 Johan Lilius
Dialectical Nets: A Categorical Approach to Net Theory. November 1989.

HUT-TCS-B9 Johan Lilius
On the Notion of Dialectical Nets. June 1990.

HUT-TCS-B10 Kari J. Nurmela, Patric R. J. Östergård
Constructing Covering Designs by Simulated Annealing. January 1993.

HUT-TCS-B11 Peter Grönberg, Mikko Tiusanen, Kimmo Varpaaniemi
PROD—A Pr/T-Net Reachability Analysis Tool. June 1993.

HUT-TCS-B12 Kimmo Varpaaniemi
On Computing Symmetries and Stubborn Sets. April 1994.

HUT-TCS-B13 Kimmo Varpaaniemi, Jaakko Halme, Kari Hiekkanen, Tino Pyssysalo
PROD Reference Manual. August 1995.

HUT-TCS-B14 Tuomas Aura
Modelling the Needham-Schröder Authentication Protocol with High Level Petri Nets.
September 1995.

HUT-TCS-B15 Eero Lassila
ReFIEx—An Experimental Tool for Special-Purpose Processor Code Generation.
March 1996.

HUT-TCS-B16 Markus Malmqvist
Methodology of Dynamical Analysis of SDL Programs Using Predicate/Transition Nets.
April 1997.

HUT-TCS-B17 Tero Jyrinki
Dynamical Analysis of SDL Programs with Predicate/Transition Nets. April 1997.

HUT-TCS-B18 Tommi Syrjänen
Implementation of Local Grounding for Logic Programs with Stable Model Semantics.
October 1998.

HUT-TCS-B19 Marko Mäkelä, Jani Lahtinen, Leo Ojala
Performance Analysis of a Traffic Control System Using Stochastic Petri Nets.
December 1998.

HUT-TCS-B20 Eero Lassila
A Tree Expansion Formalism for Generative String Rewriting. June 2001.

ISBN 951-22-5554-5
ISSN 0783-540X