A Tangled Hierarchy of Graph-Constructing Graphs

Chris Salzberg[1,2], Hiroki Sayama[1] and Takashi Ikegami[2]

[1] Department of Human Communication, University of Electro-Communications, Tokyo 182-8585, Japan
[2] Graduate School of Arts and Sciences, University of Tokyo, Tokyo 153-8904, Japan

Abstract

The traditional construction paradigm of machine and tape is reformulated in a functionally homogeneous space of directed graph structures. Hierarchy-based roles, normally appointed to actors in a construction process, are dissolved and replaced by symmetric, level-less *engagement*. The separation between static (information carrying) and active (information processing) structures, imposed by mandate of the rules or physics in earlier theoretical models, results instead purely from graph topology. While encompassing traditional machine-tape paradigms as a special case, the formalism is shown to incorporate a wider class of construction relations. Exploiting its flexibility, a representation of a Turing machine is demonstrated, establishing computation universality. The concept of a "Tangled Construction Hierarchy" is introduced.

Introduction

The formalistic study of machine construction has its roots in the self-reproducing automata theory of mathematician John von Neumann (von Neumann, 1966). Inspired by the extreme "complication" of real living organisms, von Neumann sought to realize the emergence of complexity-increasing evolution in a functionally homogeneous medium governed only by local rules. With Stan Ulam he invented a formulation for this purpose, now known as *cellular automata* (CA) and widely adopted for the modeling of self-replication (Sipper, 1998), in which the outputs and inputs of a construction process are fundamentally made of the same "stuff". Embedded in this architecture von Neumann instantiated his automata theory in the form of a complicated universal construction machine capable of self-reproduction. By introducing static self-description (tape) and division of labour (translation/transcription), he thus demonstrated a loophole in the construction paradox stating that "a machine tool is more complicated than the elements that can be made with it" (von Neumann, 1966, p.79). Decades earlier, Turing (Turing, 1936) had initiated the study of *computing* machines on the basis of a similar machine-tape paradigm; von Neumann imported it to the realm of *constructing* machines.

Hierarchical separation of machine and tape has since played an influential role in shaping the design of artificial construction and self-replication models (Mange and Sipper, 1998; McMullin, 2000), yet notable alternatives have been proposed. Hofstadter (Hofstadter, 1979), drawing inspiration from "the molecular logic of the living state" (Lehninger, 1976), blurred this separation in a typographical system he called "Typogenetics". The players in this system are "strands": strings of characters acting both as data to be manipulated and as an active "typographical enzyme" to be applied to other strands. Hofstadter called this mixing of levels a "Tangled Hierarchy" (Fig. 1), contrasting it with the case, typified by formal systems, in which there is a clear distinction between rules and the strings they apply to. Along similar lines, Laing (Laing, 1976) devised a system in which a pair of tapes undergo local sliding and state changes leading to self-reproduction via self-inspection. By means of transfer primitives, active and passive roles are arbitrarily exchangeable in this process, exemplifying a uniquely mixed-level style of execution.

All of these models assign roles in a construction process employing the intuitive concept of *levels*. Describing his system, Hofstadter asserts that: "The two-way street which links 'upper' and 'lower' levels of Typogenetics shows that, in fact, neither strand nor enzyme can be thought of as being on a higher level than the other." (Hofstadter, 1979, p.513) The word "level" is used here in the sense of *containing the same information*, interpreted as passive *data* in one case (strand), and as active *process* in the other (enzyme). von Neumann's machine contains a related type of "levels". As do strand and enzyme, the tape and machine it codes for, interpreted according to a set of transition rules[1], contain the same information: one in a passive form, the other in an active one. By analogy, they are thus — according to Hofstadter's use of the word — on the same "level".

[1] Strictly speaking an embedded universal construction machine is also required in the initial configuration to carry out translation.

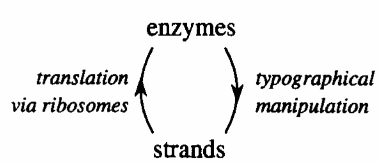

Fig. 1: The "Central Dogma of Typogenetics", an example of a Tangled Hierarchy (Hofstadter, 1979, p.513).

This concept of levels is, however, elusive: it defines two things as being on the same level if they differ only by interpretation, in one case by the human interpreter, in the other by a complicated set of CA rules. Yet this interpretation implicitly *assigns* levels: enzyme acts on strand, machine acts on tape, and never the other way around. Even in Laing's model, in which these roles are exchangeable, at any one time we interpret one tape as *active* (machine) and the other as *passive* (tape). If ultimately we aim to understand complex level-crossings and mixings of the living state, then such hierarchical interpretation is potentially misleading. Thus comes the question, the underlying theme of this paper: can we separate the functionally active *state* — that which depends on interpretation — from the structurally embedding *medium*? This requires that we replace specialized rules of *interpretation* by symmetric rules of *engagement*, and that we operate on a space that is sufficiently flexible to simultaneously accommodate both active (machine/enzyme) and passive (tape/strand) topologies.

In the sections that follow we demonstrate that these constraints are jointly satisfiable in a space of locally-interacting, collaboratively constructing graph structures in which the status of machine or tape, or the non-applicability of such labels, is an emergent and dynamic property of an *engagement* rather than an appointed *role*. By translating the output of such "level-less" engagement into the creation of new graph structures, we close the loop of the Tangled Hierarchy (Fig. 1), yet in a strictly *non-hierarchical* way. Exploiting its flexibility, we show a simple representation of a Turing machine, the quintessential top-down system, in the proposed framework. This example is used to illustrate that it is the *structure* of the graph, and not the *rules* of our formulation, that now defines machine and tape, and hence that we are working at a deeper level than any "level"-based system.

Formulation

The formulation we present in this section is conceptually divided into three parts:

(i) A *medium* of directed graph structures with input/output labels from a finite alphabet.

(ii) *Rules of engagement* according to which graph structures can read and be read by one another, producing as output a string of symbols from the alphabet of (i).

(iii) *Rules of translation* transforming the output of (ii) into an active process of growth and folding on the same graph structure (i) from which it was produced.

We introduce the medium (i) and rules (ii) by reformulating top-down input-reading of a finite-state machine as a non-hierarchical, symmetrical engagement between pointers to nodes of a directed, labeled graph. This one-on-one engagement is subsequently generalized to an arbitrary number of pointers. Finally, the translation process (iii), constituting the upward arrow of Fig. 1, is introduced, completing our formulation which we call a "Graph-Constructing Graph".

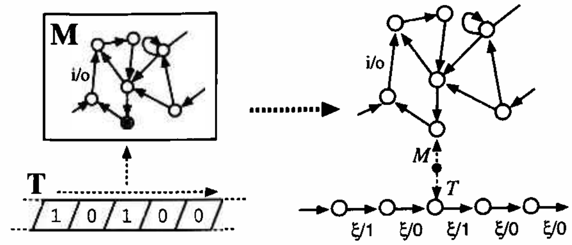

Fig. 2: Machine (**M**) and input tape (**T**) before and after symmetry is imposed.

Machines and tapes

We begin with the standard conceptual abstraction of a physical machine, that of a finite-state machine (FSM). In this paper we focus on a particular deterministic FSM called a *Mealy machine*, defined formally as a 5-tuple $\mathbf{M} = (S, I, O, \delta, \varphi)$, where S is a set of states, I an input alphabet, O an output alphabet, $\delta : I \times S \to S$ a transition function and $\varphi : I \times S \to O$ an output function. We simplify this machine by assuming a general input/output alphabet $A = I = O$, allowing us to construct composite functions from δ and φ; this will be essential for the steps that follow.

The machine **M** is a finite automaton, and in automata theory there is a clear distinction between the input/output (strings of characters from A) and the machine (a labeled directed graph). Converting input-reading, the active process in this system, from interpretation to engagement requires firstly that we reformulate the input string as a special case of a graph. For this purpose we introduce a special symbol 'ξ', inspired by the ε-transition of the nondeterministic FSM that requires no input for transition, as an element of $\Sigma = A \cup \{\xi\}$. ξ serves as a "default path" symbol, allowing us to map a linear tape to a chain of ξ/A-transitions, shown in Fig. 2 for $A = \{0, 1\}$. The set S now contains nodes (states) of both machine *and* tape, their hierarchy replaced by a *relation* between two graph structures, henceforth referred to via pointers M and T indexing states $s_M \in S$ and $s_T \in S$ respectively. Using the notation $\delta_X(\sigma) = \delta(\sigma, s_X)$ and $\varphi_X(\sigma) = \varphi(\sigma, s_X)$ for $X \in \{M, T\}$ and $\sigma \in \Sigma$, the tape-read/state-transition cycle can thus be described as:

1. T sends $\varphi_T(\xi)$ ("data") to M and advances along $\delta_T(\xi)$ (moves tape head forward).
2. M sends $\varphi_M(\varphi_T(\xi))$ to the output string and advances along $\delta_M(\varphi_T(\xi))$ (state-transition).

The above steps can be represented in terms of the function composition sequence $\xi \to T \to M$, understood as the operation: advance T on $\delta_T(\xi)$ and M on $\delta_M(\varphi_T(\xi))$ and output $\varphi_M(\varphi_T(\xi))$. This process is easily generalized to arbitrary orderings of elements from the set $\{M, T\}$, which we call *flows*. There are four of these: $(M \to T)$, (M), $(T \to M)$ and (T), drawn in Fig. 3 as directed paths on an "M/T loop". The second flow, for instance, reads as: advance M on $\delta_M(\xi)$ and output $\varphi_M(\xi)$. In general, there is a uniform procedure for each participant in these flows:

a. **Receive** output $\sigma \in \Sigma$ from the last element.

b. **Send** $\varphi_X(\sigma)$ to the next element.

c. **Advance** along the transition $\delta_X(\sigma)$.

where $X \in \{M,T\}$, $\sigma = \xi$ for the first element, and the last element's "next element" is the output string. While encompassing top-down machine/tape hierarchies in the flows $(T \to M)$ and (M) (flows that terminate with M), this procedure generalizes to more unconventional structures such as, for example, branching (graph-based) tapes. All that remains is to define, at any time, *which* flow to execute, and to do so in a way that preserves the top-down hierarchy as a special case. We do this using the following symmetric tree:

$$\begin{tikzcd} \xi \arrow[r, "\delta_X"] \arrow[dr, "\delta_Y"'] & \varphi_X(\xi) \arrow[r, "\delta_Y"] & \varphi_Y(\varphi_X(\xi)) \\ & \varphi_Y(\xi) \arrow[r, "\delta_X"] & \varphi_X(\varphi_Y(\xi)) \end{tikzcd}$$

where $\{X,Y\}$ is a permutation of the set $\{M,T\}$. Each node in this tree corresponds to the output of a possible flow; links are prioritized from top to bottom and followed if the corresponding transition function, applied to the previous output, is defined. For example, if $\delta_Y(\xi)$ and $\delta_X(\varphi_Y(\xi))$ are defined but not $\delta_X(\xi)$, then we choose the lower branch and return the flow $(Y \to X)$ with output $\varphi_X(\varphi_Y(\xi))$. If no ξ-transition exists for any $X \in \{M,T\}$ then we say that the system has reached a *stopping configuration*. In any other case the flow returned is called the *ξ-flow*.

The steps described so far maintain symmetry in engagement: both machine and tape structures are treated in the same way. To complete the formulation and maintain this symmetry, the permutation $\{X,Y\}$ in the preceding tree must be fixed in an accordingly non-hierarchical way. We could, for instance, stipulate that $X \equiv M$, yet this would explicitly distinguish M and T by prioritizing flows $(M \to T)$ and (M) over $(T \to M)$ and (T). Instead, we opt to treat X as a dynamic *lead element*, defined as the *last* element in the *previous* ξ-flow. The ordering $\{X,Y\}$ of branches in the tree is thus also dynamic: following ξ-flows (M) and $(T \to M)$ the order is $\{M,T\}$, whereas following (T) and $(M \to T)$ the order is $\{T,M\}$. Applied to an engagement between two structures (call them "machine" and "tape") with transitions exclusively of the form σ/a and ξ/a, respectively, for $a \in A$ and $\sigma \in \Sigma$, and initiated with M pointing to the "machine" and T to the "tape", this scheme preserves top-down hierarchy since only the first two flows ever occur (Fig. 4). Yet rather than being a result of *rules*, this hierarchy results from the *structure* we have imposed. We could, alternatively, have allowed branching on the tape; output would then result from collaborative, role-changing interaction, a kind of *co-reading* process. In the latter context, "machine" and "tape", and the tape-reading process that distinguishes them, are hard to find: their structures are reading *each other*.

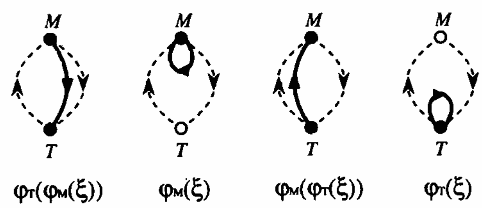

Fig. 3: Possible ξ-flows (solid arrows) and their output functions on the 2-element *M*/*T* loop (dashed arrows).

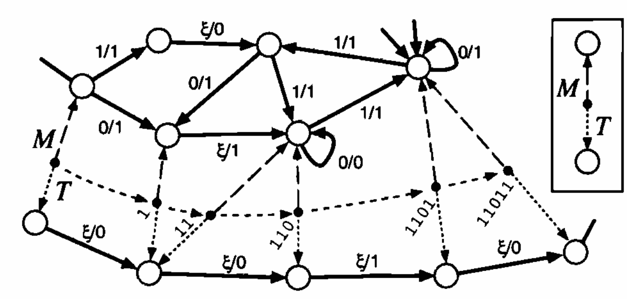

Fig. 4: Tape-reading process for machine/tape structures. Motion of *M*/*T* pointers drawn as dashed horizontal arrows. Trailing binary sequence indicates current output string at each step.

Loops and flows

The idea of co-reading can quite readily be generalized to an arbitrary number of elements. Consider the 2-element loop of Fig. 3 on the set $\{M,T\}$ a special case of an N-element loop (or N-loop) $L(t)$ on the sequence $\{s_i(t)\}_{i=1}^N \in S$, with t representing a discrete time parameter. As for the *M*/*T* system, define a lead element $l(t) \in \{1,\ldots,N\}$ and denote as L_0 the cyclic permutation of L with s_l as first element. A flow F on an N-loop L is a sequence of indices $\{f_i\}_{i=1}^J \in \{1,\ldots,N\}$ with output given by the composite function:

$$\Phi_F \quad = \quad \varphi_{f_J} \cdot \varphi_{f_{J-1}} \cdot \cdots \cdot \varphi_{f_2} \cdot \varphi_{f_1} \tag{1}$$

As an example, the set of 15 possible flows for a 3-loop is shown in Fig. 5. Execution of these flows proceeds as defined earlier for the *M*/*T* system. Beginning with $\sigma = \xi$ and ending with the output string, each participant $X \in L$

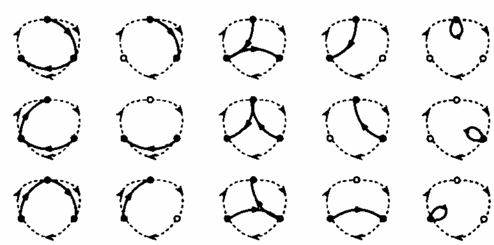

Fig. 5: Possible flows (solid arrows) on a 3-loop (dashed arrows).

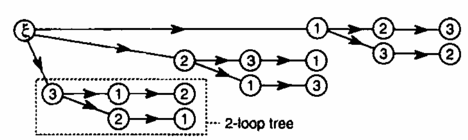

Fig. 6: Possible flows for a 3-loop $\{s_1, s_2, s_3\}$ with lead $l = 1$ as nodes of a decision tree.

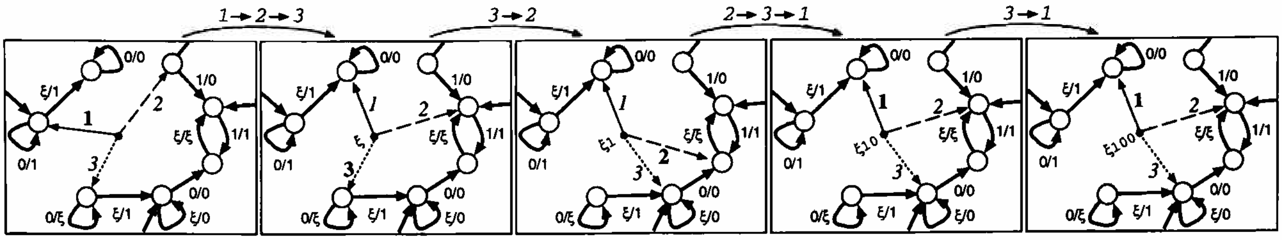

Fig. 7: Co-reading of graph structures via flows on a 3-loop. Boldface indicates lead element, trailing character sequence is output string at each step. ξ-flows drawn above frames as arrows and index sequences.

receives $\sigma \in \Sigma$ from the previous element, *sends* $\varphi_X(\sigma)$ to the next element, and *advances* along $\delta_X(\sigma)$. The decision tree of the *M/T* system also generalizes, in a recursive way: an N-loop tree is composed of N permutations of (N-1)-loop trees, is composed of (N-1) permutations of (N-2)-loop trees, etc. To give s_l priority in this tree, branches from the root are ordered according to L_0. As an example, the tree for the 3-loop of Fig. 5 is shown in Fig. 6. Each node in this tree represents a possible flow, the ξ-node being the stopping configuration. A recursive flow-finding procedure searches depth-first, prioritizing higher branches, until it reaches a terminal branch or has no further transitions available. The branch it returns is the ξ-flow, referred to as $F_\xi(L)$.

With dynamics described by F_ξ we can express the time-dependence of $L(t) = \{s_i(t)\}_{i=1}^{N}$. For each $f_j \in F_\xi(L(t))$, the state $s(f_j)(t) = s_{f_j}(t)$ indexed by f_j is updated as:

$$\begin{aligned} s(f_1)(t+1) &= \delta_{s(f_1)(t)}(\xi) \\ s(f_j)(t+1) &= \delta_{s(f_j)(t)}(\Phi_{\{f_k\}_{k=1}^{j-1}}(\xi)) \end{aligned} \qquad (2)$$

All elements not encountered in the ξ-flow are stationary, and the lead element is updated as $l(t+1) = f_J$ (the last element of the flow). An example of an execution process occurring via a 3-loop is shown in Fig. 7. The output string generated by this process is the non-terminating sequence 'ξ100000...', an example of *non-stopping* co-reading.

Translation

In leveling the rules of engagement between machine and tape, we have twisted the downward arrow of Hofstadter's "Central Dogma" (Fig. 1), the manipulation of strand (tape) by enzyme (machine), onto itself. The status of either *reading* (machine) or of *being read* (tape) is now relative, not to the rules of the system, but to the participants in an engagement. Translation, formerly the basis of equivalence between these two forms, is now something subtly different. It is a tranformation of the output of an engagement, as we have phrased it a string of symbols from Σ, *back* into the medium itself, in this case graph structure, as a growth and modification process. This can be abstractly represented as:

$$\mathbf{T}(\text{engagement}) = \text{process on structure}$$

This collaborative cycle of structure creation and modification, based on Hofstadter's Tangled Hierarchy, we call a "Tangled Construction Hierarchy". Though conceptually inspired by complex manipulation and translation processes of the living cell, unlike machine-tape models it is not *intrinsically* hierarchical: engagement is not a structure, but a process *between* structures. In the same way, the output of **T** is not exclusively structure but a process *on* structure (Fig. 8). We draw on the analogy of complex folding from amino acid sequences into proteins, represented rather simplistically in Typogenetics as 90° "kinks" in an otherwise directionless strand. In contrast, and also differing from formal graph grammars (Rozenberg, 1997) and the recently-developed graph automata (Tomita et al., 2002), our goal is that growth and folding of the graph, as of the amino acid sequence, be locally-propagating and incremental. The following properties are thus demanded of **T**: it should (1) act locally, (2) generate a minimal set of "graph-manipulation" instructions, (3) function on a minimal set of rules and (4) be construction universal in the sense that, given an appropriate output string, it should generate any arbitrary graph structure. The framework and set of rules we propose in this section achieves a desired balance between these constraints. Other possibilities, including non-Turing-computable transformations, are also being considered.

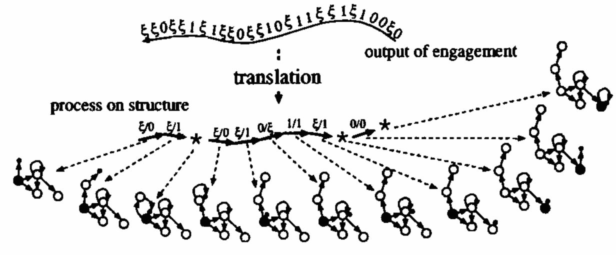

Fig. 8: Translation mapping output from engagement into process of graph growth and folding.

We assume to start only singly-labeled links; incorporation of input/output labels follows. Such that our graph-modification process is local (constraint (1)), we employ a dynamic *construction head*, loosely related to the write-head of machine-tape formalisms, acting in conjunction with a *root* pointer. The head and root can thought of as the "hands" of construction, with the root referencing a re-assignable position on the graph to which the head, moving about, carrying and affixing chains of nodes and links, can return during construction. Neither head nor root makes use of static global referencing; to modify a portion of the graph, the head, starting from either its current position or the root node, must travel along intervening links to get to it.

Only two types of graph-manipulation instructions (constraint (2)) are understood by the head: "insert/traverse link" (denoted by any link label) and "return" (denoted by '*'). A minimal set of nine rules (constraint (3)) are required for universality (constraint (4)), pictured in Fig. 9 in terms of the current instruction (link or return) and neighbourhood of the head node (node pointed to by the construction head). Depending on local topology of the graph, an instruction is interpreted by the head in one of three ways: * takes it back to the root (column 1 rules), whereas a link 'a' inserts/sprouts a new link (column 2) if the a-transition does not yet exist, or traverses the link (column 3) if it does. Note that the head can directly "hold" at most one link at a time, but can append complete *chains* of links by inserting and dragging them. Specifically, there is always a succession of *move*, *insert*, *return*, *drag* and *affix* rules that suffices to create a path connecting any accessible potions of the graph. Hence any graph having at least one node with paths to all other nodes is constructible; this is the particular form of construction universality that is achieved. Examples of strings translated and applied to a single node are given in Fig. 10.

We extend the above framework to input/output pairs by treating the pair (σ_i, σ_o) in the rules of Fig. 9 as the link instruction σ_i, with the exception that any existing output σ_o is overwritten each time a link is traversed. In addition to augmenting structure, the head can thus also modify existing links, with the exception of removing them.

Lastly, we require a mapping from the space of symbol strings (strings from the alphabet Σ) to the space of instruc-

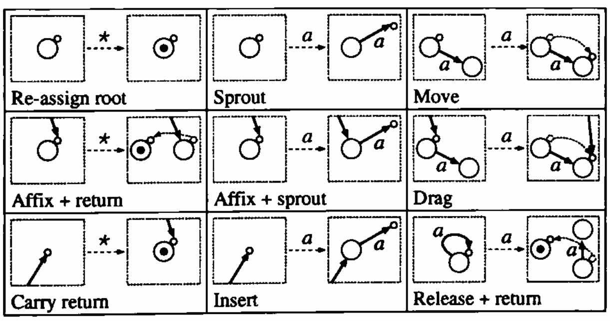

Fig. 9: Graph-growing rules. Solid arrows are links, dashed arrows are rules, 'a' is any link label, * is the return symbol, ○ is a node, ∘ is the head position, and ⊙ is the root node. Dotted arrows indicate change in position of construction head. A link pointing to directly to the head (→∘) is currently being held by the head. In the "re-assign root" rule, the root pointer is re-assigned to the current position of the head (the head itself does not move).

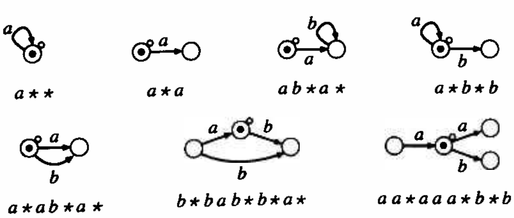

Fig. 10: Graphs produced by translating output strings and applying the resulting construction process to a single starting node.

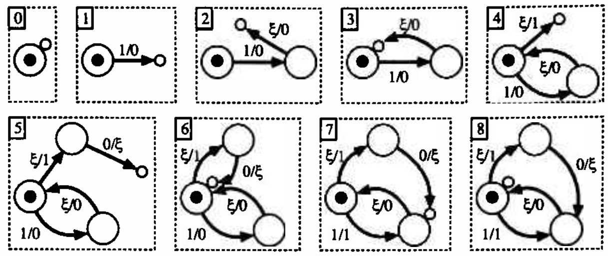

Fig. 11: Transformation of the output string 10_1 $\xi\xi 0_2$ $\xi 0_3$ $\xi\xi 1_4$ $0\xi_5$ $\xi 1_6$ 11_7 $\xi 0_8$ via codons of eq. (3) into graph growth and folding. Subscript references frame index of sequence.

tions (links (σ_i, σ_o) plus *), analogous to Hofstadter's "Typogenetic Code" (Hofstadter, 1979) inspired by the Genetic Code contained in tRNA molecules of a biological cell. This mapping, shown as the downward arrow in Fig. 8, can also be thought of as the translation between "machine code" (symbol string) and "assembly code" (growth and folding instructions). Its specification determines the type of topologies that are likely to grow. To be *universally expressible*, all possible link pairs plus * must be represented; this amounts to requiring $|\Sigma|^2 + 1$ different "codons" (substrings) in our coding scheme. One possibility is the following mapping:

$$a\sigma \rightarrow a/\sigma \qquad \xi a \rightarrow * \qquad \xi\xi\sigma \rightarrow \xi/\sigma \qquad (3)$$

where $a \in A$ and $\sigma \in \Sigma$. A transformation process from output string to construction process via this mapping is shown in Fig. 11 for the alphabet $A = \{0, 1\}$. Note that ambiguity arises in the translation process if the construction head modifies structure indexed by an element of a loop (e.g. M or T in the 2-loop), or, in the case of concurrent construction, if two heads work on the same structure. Such issues are resolvable by stipulating rules for conflict resolution, for instance by disallowing construction on indexed nodes or traversal of links on a free construction thread. Neither issue is essential to the basic formalism we have described, hence we leave open the choice of how to resolve them.

The formulation we have finished describing we call a "Graph-Constructing Graph" (GCG), pictured in its entirety in Fig. 12. As does Hofstadter's "Central Dogma" (Fig. 1) and the molecular biology from which derives its inspiration, the GCG has at its core an information creation/translation cycle: interaction of structures (graphs) produces an output (strings) that is translated to the creation (growth) and modification (folding) of these same structures. Yet whereas in Typogenetics the interaction is between passive "data" and active "process", here such dis-

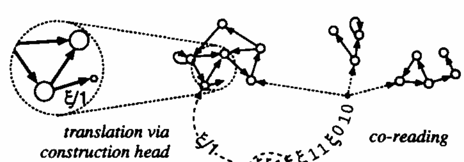

Fig. 12: The Graph-Constructing Graph, an example of a "Tangled Construction Hierarchy".

tinctions, if they exist, are entirely context-dependent. In that level-crossing and mixings are central to dynamical processes in a cell, we are thus conceptually closer to the constructing lifeforms we ultimately aim to understand.

Turing machine as GCG

The architecture of a Turng Machine as a GCG, using the codons of eq. (3) for translation, is shown in Fig. 13. Since we are only concerned with computation (reading and overwriting) and not construction (modification of graph connectivity), the return symbol '*' is not required; the root pointer thus plays no role, and is not included. We consider TM transitions of the form (a,b,c,d), where $\{a,b\} \in \{0,1\}$ are the old and new values on the tape, respectively, $c \in \{L,R\}$ is the direction to advance the write head, and $d \in \{L,R\}$ is the direction to advance the read head.[2] We use the alphabet $A = \{0,1\}$, with left and right symbols of the TM mapping to 0 and 1, respectively. Initially M points to a branch node on the machine, and T (read head) and H (write head) point to nodes on the tape. T begins as the lead element.

The key feature of this representation, when compared with Fig. 4, is that we have enabled 2-way motion on the tape: from a given node on the tape structure, the read head (T) may either read the data value (ξ/a), follow the left branch ($0/\xi$), or follow the right branch ($1/\xi$). ξ-transitions initiate the flow, so T (the lead element) begins by following the ξ/a transition, which moves M on the a/ξ branch. Since M has no ξ-transition, T leads again and the $\xi/a \rightarrow a/\xi$ flow is repeated. T follows ξ/a a third time, this time advancing M on the a/b branch. At this point the construction head (denoted as H) has received the codon $\xi\xi b$, instructing it to overwrite the ξ/a link with the new data value ξ/b.

T and M have not yet advanced, however. M still has no ξ-transition, so T follows ξ/b and M follows b/c, outputting c to H. Now M has a ξ-transition, ξ/d, which it follows. The value of d determines in which direction T will advance (read head move), but in both cases ξ is sent to H. The codon $d\xi$ translates to d/ξ, which advances H along d (write head move). This completes the basic read/write cycle of the Turing Machine. Assuming an infinite tape, we have achieved TM-equivalence and hence computation universality.

[2]The option of moving the read and write head in different directions is included as it emerges naturally from the reformulation.

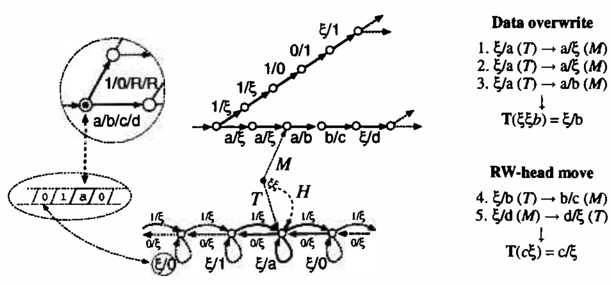

Fig. 13: Turing machine as GCG (left) and flows comprising a single read/write operation (right). a,b,c,d represent symbols from $\{0,1\} \equiv \{L,R\}$. In this picture the TM has just completed flow 2.

Note that we have significantly handicapped the GCG to show TM-equivalence. It is quite straightforward, for instance, to build a TM with alternate structures for data storage (e.g. 2D/3D tape) or one with finite but extendible tape (as done in (Tomita et al., 2002)). Such topics will be discussed with more detail in a follow-up publication.

Conclusion

In this paper we introduced a formulation motivated by von Neumann's original search for the logic of construction. Using as medium a space of directed graph structures, a system of Graph-Constructing Graphs was described. This system, an example of what we call a "Tangled Construction Hierarchy", is ultimately aimed at abstracting tangled level-crossings of hierarchy in biological systems to a finite space of logical structures (i.e. graphs) with discrete dynamics. The underlying theme we are tackling is that of the essential or non-essential nature of machine-tape dualities in construction, and of the possibility for potentially revealing alternatives. In so doing, we are striving for a new perspective on information creation and manipulation in constructing systems. That hierarchical segregation is so patently violated by biology's most central constructing system, the cell, makes such a perspective relevant and necessary.

An immediate future goal of this work is to explore the space of graph structures and their interactions to search for fixed points in cycles of construction. Evolutionary dynamics in a "soup" of GCGs is one possible environment for this exploration; others are being considered.

Acknowledgments

C.S. acknowledges financial support by grants from the Netherlands Organization for International Cooperation in Higher Education (Nuffic) and the VSB Funds.

References

Hofstadter, D. R. (1979). *Gödel, Escher, Bach: an Eternal Golden Braid*, chapter XVI, pages 504–513. Basic Books, Inc.

Laing, R. (1976). Automaton introspection. *Journal of Computer and System Sciences*, 13(2):172–183.

Lehninger, A. (1976). *Biochemistry*. Worth Publishers, New York.

Mange, D. and Sipper, M. (1998). Von Neumann's quintessential message: genotype + ribotype = phenotype. *Artificial Life*, 4:225–227.

McMullin, B. (2000). John von Neumann and the evolutionary growth of complexity: Looking backward, looking forward... *Artificial Life*, 6:347–361.

Rozenberg, G., editor (1997). *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1 of *Foundations*. World Scientific, Singapore.

Sipper, M. (1998). Fifty years of research on self-replication: An overview. *Artificial Life*, 4:237–257.

Tomita, K., Kurokawa, H., and Murata, S. (2002). Graph automata: natural expression of self-reproduction. *Physica D*, 171(4):197–210.

Turing, A. M. (1936). On computable numbers with an application to the entscheidungsproblem. *Proc. London Math. Soc. Ser.*, 42:230–265.

von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, Illinois. Edited and completed by A. W. Burks.