New Book:

Grammars with Context Conditions
and Their Applications

Alexander Meduna and Martin Švec

Wiley, New Jersey, 2005

http://www.fit.vutbr.cz/~meduna/books/gwcc/

ISBN 0-471-71831-9

Outline

1. **Introduction**
   Classification of grammars with context conditions.
2. **Grammars with Conditions Placed on Derivation Domains**
   Grammars whose direct derivations are defined over a word monoid.
3. **Grammars with Conditions Placed on the Use of Productions**
   Grammars whose productions are applicable on the condition that certain substrings occur or do not occur in the rewritten sentential form.
4. **Grammas with Conditions Placed on the Neighborhood of Rewritten Symbols**
   Continuous-context and scattered-context grammars and their uniform versions.
5. **Derivation Simulation**
   A formalization of grammatical derivation similarity.
6. **Applications**
   Applications of grammars with context conditions in biology.

1. Introduction

The classical language theory is based on the Chomsky hierarchy of regular, context-free, context-sensitive, and phrase-structure grammars.
These grammars have several disadvantages concerning the contextual sensitivity:

Disadvantages of Regular and Context-Free Grammars

- no context sensitivity
- significantly less powerful than context-sensitive and phrase-structure grammars

Disadvantages of Context-Sensitive and Phrase-Structure Grammars

- strict conditions placed on the context surrounding the rewritten symbol
- complex form of rewriting productions
- difficult to use in practice

Advantages of Grammars with Context Conditions

- they are based on context-independent productions
- their context conditions are simple and flexible
- they are as powerful as classical context-dependent grammars

Classification of Their Context Conditions

- conditions placed on derivation domains
- conditions placed on the use of grammatical productions
- conditions placed on the neighborhood of the rewritten symbols

Sequential and Parallel Conditional Grammars

Both sequential and parallel versions of the grammars are studied. Sequential grammars rewrite only one symbol during a derivation step while parallel grammars rewrite all symbols in one derivation step.

A Context-Free Grammar is a quadruple $G = (V, T, P, S)$, where

- V is the total alpabet
- T is a finite set of terminal symbols, $T \subset V$
- P is a finite set of productions $A \rightarrow x$, where $A \in V - T$, $x \in V^*$
- S is the axiom, $S \in V - T$

Context-Free Derivation

Given uAv and uxv, where $A \in V - T$, $u, v, x \in V^*$,

$$uAv \Rightarrow_G uxv \text{ if and only if } A \rightarrow x \in P$$

Notation

- **CF** – the family of context-free langages
- **CS** – the family of context-sensitive languages
- **RE** – the family of recursively enumerable languages

Classical Parallel Grammars

An E0L Grammar is a quadruple $G = (V, T, P, S)$, where V, T, and S have the same meaning as in context-free grammars.

Productions of E0L grammars have the form $a \rightarrow x$, where $a \in V$, $x \in V^*$.

Derivation in E0L Grammars

Given $u = a_1 \ldots a_n$ and $v = x_1 \ldots x_n$,

$$\boxed{u \Rightarrow_G v \text{ if and only if } a_i \rightarrow x_i \in P \text{ for all } i = 1, \ldots, n}$$

Illustration

$$\begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & \ldots & a_n \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow \\ x_1 & x_2 & x_3 & x_4 & x_5 & \ldots & x_n \end{array}$$

Families of languages

- **E0L** – the family of languages generated by E0L grammars
- **EP0L** – E0L grammars without erasing productions (propagating E0L grammars)

Generative power of E0L grammars

$$\mathbf{CF} \subset \mathbf{E0L} = \mathbf{EP0L} \subset \mathbf{CS}$$

2. Conditions Placed on Derivation Domains

- Classical grammars define the derivation relation over V^*, where V is an alphabet.
- Grammars over word monoids define the derivation relation over W^*, where W is a finite language.

Basic idea

$$V = \{a, b, c, d\}$$
$$W = \{a, bc, d\}$$

$aababd \in V^*$ – can be obtained by a concatenation of members of V;

$aababd \notin W^*$ – cannot be obtained by a concatenation of members of W;

$aabcad \in W^*$ – can be obtained by a concatenation of a, a, bc, a, d from W;

Intuitively, the string bc in W represents a context condition "b's right neighbor has to be c".

Context-Free Grammars over Word Monoids

Context-free grammar over word monoid (*wm*-grammar) is a pair (G, W)

- $G = (V, T, P, S)$ is a context-free grammar
- W is a finite language over V
- degree of (G, W) is the maximal length of strings in W

The relation of a direct derivation from u to v is defined as

$$u \Rightarrow_{(G,W)} v \text{ if and only if } u \Rightarrow_G v \text{ and } u, v \in W^* \text{(!)}$$

Families of languages

- **WM**(i), **WM** – families of languages generated by *wm*-grammars of degree i and of any degree, respectively
- **prop-WM**(i), **prop-WM** – no erasing productions are allowed

Generative power

$$\textbf{prop-WM}(1) = \textbf{WM}(1) = \textbf{CF}$$
$$\subset$$
$$\textbf{prop-WM}(2) = \textbf{prop-WM} = \textbf{CS}$$
$$\subset$$
$$\textbf{WM}(2) = \textbf{WM} = \textbf{RE}$$

Reduction of *wm*-grammars

Theorem: Every $L \in$ **RE** can be defined by a ten-nonterminal context-free grammar over a word monoid generated by an alphabet and six words of length two.

E0L grammar over word monoid (WME0L grammar) is a pair (G, W)

- $G = (V, T, P, S)$ is an E0L grammar
- W is a finite language over V
- degree of (G, W) is the maximal length of strings in W

The relation of a direct derivation from u to v is defined as

$$u \Rightarrow_{(G,W)} v \text{ if and only if } u \Rightarrow_G v \text{ and } u, v \in W^* \text{(!)}$$

Families of languages

- **WME0L**(i), **WME0L** – families of languages generated by WME0L grammars of degree i and of any degree, respectively
- **WMEP0L**(i), **WMEP0L** – no erasing productions are allowed

Generative power

$$\textbf{CF}$$
$$\subset$$
$$\textbf{WMEP0L}(1) = \textbf{WME0L}(1) = \textbf{EP0L} = \textbf{E0L}$$
$$\subset$$
$$\textbf{WMEP0L}(2) = \textbf{CS}$$
$$\subset$$
$$\textbf{WME0L}(2) = \textbf{RE}$$

A Context-Conditional Grammar is a grammar with context conditions represented by strings associated with productions.

Types of conditions

forbidding – the string must not occur as a substring of the sentential form

permitting – the string must occur in the sentential form

A production is applicable to a sentential form if each of its permitting conditions occurs in the sentential form and any of its forbidding conditions does not.

Example

$S \Rightarrow^* ABCD \Rightarrow ?$

$(A \rightarrow x, \emptyset, \{C\})$ – cannot be applied: C occurs in $ABCD$

$(A \rightarrow y, \{AB\}, \{F\})$ – can be applied: AB is in $ABCD$, F is not in $ABCD$

Sequential Context-Conditional Grammars

A Sequential Context-Conditional Grammar is a context-free grammar with sets of permitting and forbidding conditions attached to productions.

Productions have the form

$$(A \to x, Per, For)$$

- Per – finite set of permitting conditions, $Per \subseteq V^+$.
- For – finite set of forbidding conditions, $For \subseteq V^+$.

Such a production can rewrite A to x provided that all strings from Per occur in the sentential form and no string from For occurs in the sentential form

Degree

A context-conditional grammar has degree (r, s) if the maximal length of permitting conditions is less or equal r and if the maximal length of forbidding conditions is less or equal s.

Sequential Context-Conditional Grammars (#2)

Families of Languages

- Context-conditional grammars with erasing productions generate **RE**.
- Context-conditional grammars without erasing productions generate exactly **CS**.

Random-Context Grammars

- permitting conditions are nonterminal symbols
- no forbidding conditions
- $(A \rightarrow x, Per, For)$, $Per \subseteq N$, $For = \emptyset$
- introduced by A. P. J. van der Walt, 1970

Random-Context Grammars with Appearance Checking

- permitting and forbidding conditions are nonterminal symbols
- $(A \rightarrow x, Per, For)$, $Per, For \subseteq N$

Forbidding Grammars

- no perimitting conditions
- forbidding conditions are nonterminal symbols
- $(A \rightarrow x, Per, For)$, $Per = \emptyset$, $For \subseteq N$
- M. Penttonen, 1975

A Generalized Forbidding Grammar (*gf*-grammar) is a context-conditional grammar in which every production contains no permitting conditions.

$$\boxed{(A \rightarrow x, Per, For) \text{ satisfies } Per = \emptyset}$$

Families of Languages

- **GF**(i) and **GF** – families of languages generated by *gf*-grammars of degree i and of any degree, respectively
- **prop-GF**(i) and **prop-GF** – no erasing productions are allowed

Generative Power

$$\textbf{prop-GF}(0) = \textbf{GF}(0) = \textbf{CF}$$
$$\subset$$
$$\textbf{GF}(1) = \textbf{F}$$
$$\subset$$
$$\textbf{GF}(2) = \textbf{GF} = \textbf{RE}$$

Reduction of *gf*-grammars

Theorem: Every $L \in$ **RE** can be defined by a generalized forbidding grammar of degree 2 with no more than 13 conditional productions and 15 nonterminals.

Simple Semi-Conditional Grammars

A Simple Semi-Conditional Grammar (*ssc*-grammar) is a context-conditional grammar in which every production contains no more than one condition.

$$(A \rightarrow x, Per, For) \text{ satisfies } |Per| + |For| \leq 1$$

Families of Languages

- **SSC**(r, s), **SSC** – families of languages generated by *ssc*-grammars of degree (r, s) and of any degree, respectively.
- **prop-SSC**(r, s), **prop-SSC** – no erasing productions are allowed

Generative Power

$$\textbf{CF}$$
$$\subset$$
$$\textbf{prop-SSC} = \textbf{prop-SSC}(2,1) = \textbf{prop-SSC}(1,2) = \textbf{CS}$$
$$\subset$$
$$\textbf{SSC} = \textbf{SSC}(2,1) = \textbf{SSC}(1,2) = \textbf{RE}$$

Reduction of *ssc*-grammars

Theorem: Every $L \in$ **RE** can be defined by a simple semi-conditional grammar of degree (2,1) with no more than 12 conditional productions and 13 nonterminals.

E(T)0L Grammars

- an important type of L-systems (A. Lindenmayer)
- based on context-independent productions
- T = several sets of productions
- all symbols are simultaneously rewritten during a derivation step:

$$\begin{array}{cccccccc} a & a & B & a & A & c & A & d \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \end{array}$$

Generative Power of E(T)0L Grammars

CF $\subset$ **E0L** $\subset$ **ET0L** $\subset$ **CS**

A Context-Conditional ET0L Grammar is an ET0L grammar with sets of permitting and forbidding conditions attached to productions.

Productions

$(a \rightarrow x, Per, For)$

- Per – finite set of permitting conditions, $Per \subseteq V^+$;
- For – finite set of forbidding conditions, $For \subseteq V^+$.

Such a production can rewrite a to x provided that all strings from Per occur in the sentential form and no string from For occurs in the sentential form

Degree (r, s)

Defined by analogy with the degree of sequential context-conditional grammars; that is, r is the maximal length of a permitting condition in Per and s is the maximal length of a forbidding condition in For.

Forbidding ET0L Grammars (FET0L grammars) are context-conditional ET0L grammars with productions having only forbidding context conditions.

$$(a \to x, Per, For) \text{ satisfies } Per = \emptyset$$

Degree

The maximal length of forbidding context conditions.

Families of Languages - notation

- **F** = forbidding conditions
- **T** = several sets of productions
- **P** = without erasing productions (propagating)

Generative Power

$$\mathbf{CF}$$
$$\subset$$
$$\mathbf{FEP0L}(0) = \mathbf{FE0L}(0) = \mathbf{EP0L} = \mathbf{E0L}$$
$$\subset$$
$$\mathbf{FEP0L}(1) = \mathbf{FEPT0L}(1) = \mathbf{FE0L}(1) = \mathbf{FET0L}(1) =$$
$$\mathbf{FEPT0L}(0) = \mathbf{FET0L}(0) = \mathbf{EPT0L} = \mathbf{ET0L}$$
$$\subset$$
$$\mathbf{FEP0L}(2) = \mathbf{FEPT0L}(2) = \mathbf{FEP0L} = \mathbf{FEPT0L} = \mathbf{CS}$$
$$\subset$$
$$\mathbf{FE0L}(2) = \mathbf{FET0L}(2) = \mathbf{FE0L} = \mathbf{FET0L} = \mathbf{RE}$$

A Simple Semi-Conditional ET0L Grammar (SSC-ET0L grammar) is a context-conditional ET0L grammar in which every production contains no more than one context condition.

$$(a \to x, Per, For) \text{ satisfies } |Per| + |For| \leq 1$$

Families of Languages

Denoted by analogy with forbidding ET0L grammars; however, instead of prefix **F**, we use **SSC-** in terms of SSC-ET0L grammars.

Generative Power

$$\textbf{CF}$$
$$\subset$$
$$\textbf{SSC-EP0L}(0,0) = \textbf{SSC-E0L}(0,0) = \textbf{EP0L} = \textbf{E0L}$$
$$\subset$$
$$\textbf{SSC-EPT0L}(0,0) = \textbf{SSC-ET0L}(0,0) = \textbf{EPT0L} = \textbf{ET0L}$$
$$\subset$$
$$\textbf{SSC-EP0L}(1,2) = \textbf{SSC-EPT0L}(1,2) = \textbf{SSC-EP0L} = \textbf{SSC-EPT0L} = \textbf{CS}$$
$$\subset$$
$$\textbf{SSC-E0L}(1,2) = \textbf{SSC-ET0L}(1,2) = \textbf{SSC-E0L} = \textbf{SSC-ET0L} = \textbf{RE}$$

Continuous Context Grammars

Represented by classical context-sensitive and phrase-structure grammars. They are based on productions of the form

$$uAv \rightarrow uxv$$

The strings u and v can be interpreted as continuous context conditions.

Scattered Context Grammars rewrite several symbols in one step.

The rewritten symbols can be scattered across the sentential form; however, all of them must occur in the sentential form and they must occur in the prescribed order. Thus, they can be interpreted as scattered context conditions.
Example: consider a production $(B, C, D) \rightarrow (x, y, z)$. Then,

$$\begin{array}{ccccccccc} C & B & A & B & C & c & D & A & d \\ & \downarrow & & & \downarrow & & \downarrow & & \\ C & x & A & B & y & c & z & A & d \end{array}$$

Aim

To make the generation of languages by phrase-structure grammars more uniform.

Results

Statement: For every phrase-structure grammar, there exists an equivalent phrase-structure grammar having the sentential forms based on a sequence of substrings, each of which represents a permutation of symbols over a very small alphabet.

Analogical results were achieved for

- phrase-structure grammars
- EIL grammars
- scattered context grammars

Reduction of Scattered Context Grammars

Scattered context grammars can be reduced with respect to the

- number of nonterminals
- degree of context sensitivity: number of context-sensitive productions
- maximal context sensitivity: maximal length of context-sensitive productions
- total context sensitivity: overall length of context-sensitive productions

Results

- three-nonterminal scattered context grammars generate **RE**
- several simultaneous reductions of complexity measures

5. Derivation Simulation

Some equivalent formal language models generate their language in a more similar way than others. Is it possible to formalize this similarity? Which transformations of language models satisfy this intuitive understanding of simulation?

Goals

- formalization of the similarity of rewriting processes
- more detailed approach to the equivalency of formal models
- new transformations leading to models closely simulating each other

Context-free grammars G_1 and G_2:

G_1: $S \rightarrow aSb$, $S \rightarrow ab$

G_2: $S \rightarrow aB$, $B \rightarrow Ab$,
$A \rightarrow aB$, $B \rightarrow b$

$$L(G_1) = L(G_2) = \{a^n b^n : n \geq 1\}$$

σ is a suitable substitution from G_2's alphabet to G_1's alphabet.

For every derivation step in G_1, there are no more than two corresponding derivation steps in G_2. We say that G_2 2-closely simulates G_1 with respect to σ.

$$
\begin{array}{ccc}
G_1 & \xleftarrow{\quad\sigma\quad} & G_2 \\
\hline
S & \xleftarrow{S \in \sigma(S)} & S \\
 & & \Downarrow \\
\Downarrow & & aB \\
 & & \Downarrow \\
aSb & \xleftarrow{aSb \in \sigma(aAb)} & aAb \\
 & & \Downarrow \\
\Downarrow & & aaBb \\
 & & \Downarrow \\
aaSbb & \xleftarrow{aaSbb \in \sigma(aaAbb)} & aaAbb \\
\Downarrow & & \Downarrow \\
\vdots & & \vdots \\
\Downarrow & & \Downarrow \\
a^n b^n & \xleftarrow{a^n b^n \in \sigma(a^n b^n)} & a^n b^n
\end{array}
$$

Results

- general formal model of derivation simulation
- several special cases: n-close simulation, homomorphic simulation, ...
- formalization of grammatical simulations

Practical Example

Theorem: For every E(0,1)L grammar G, there exists an equivalent WME0L(2) grammar G' and a homomorphism ω such that G' 1-closely homomorphically simulates G with respect to ω.

Three case studies in terms of a biological simulation.

Case Studies

- stagnation of a cellular organism infected by a virus
- degeneration and death of a red alga
- plant simulation controlled by a resource flow

Virus-Infected Cellular Organism

This case study demonstrates development of a simple cellular organism infected by a virus. During a healthy development, every cell of the organism divides itself into two cells. However, when a virus infects some cells, all the organism stagnates until it is cured again.

Model

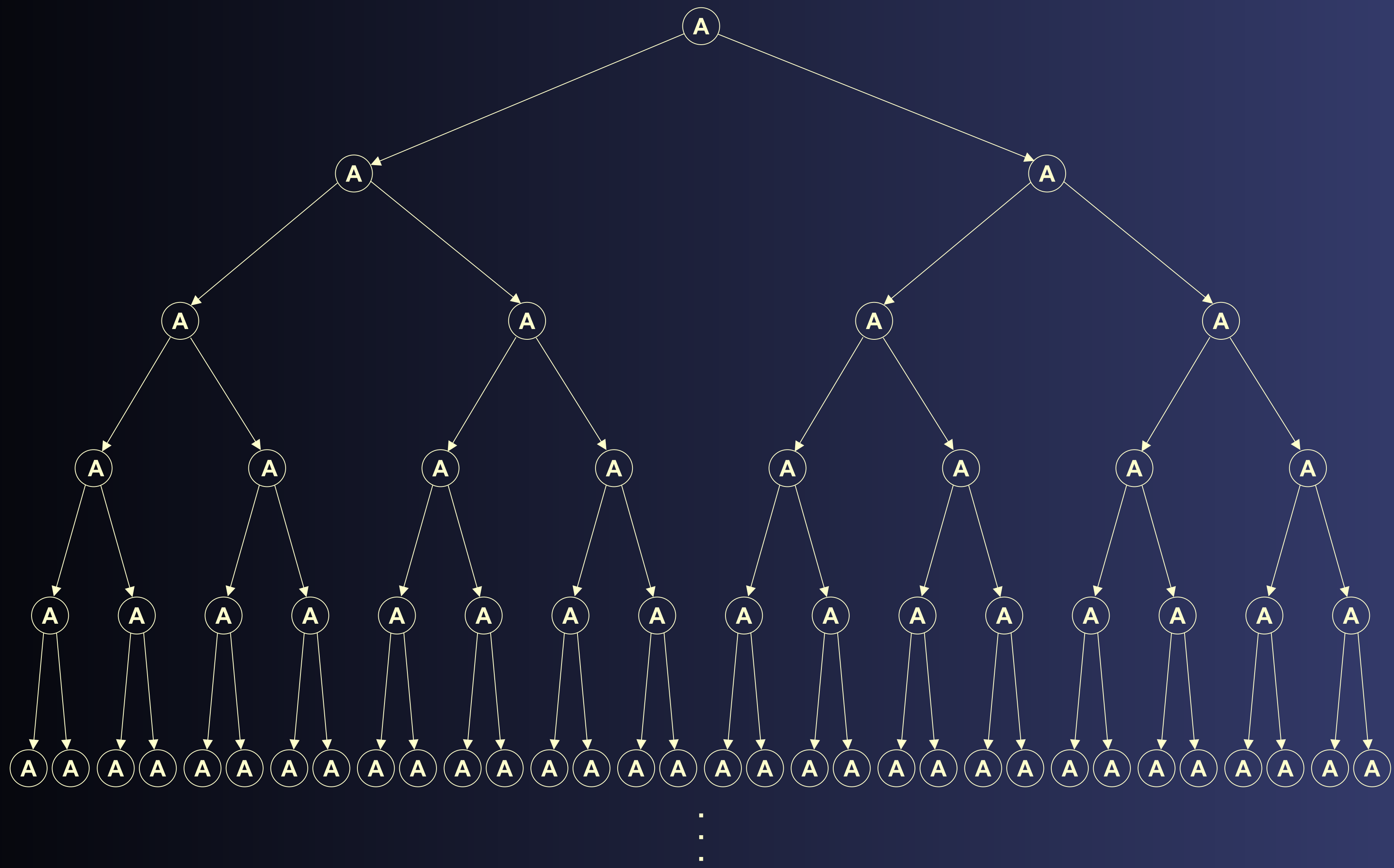
Consider a simple semi-conditional 0L grammar, G, with productions:

$$\begin{array}{ll} (A \rightarrow AA, \emptyset, \{B\}), & (A \rightarrow AB, \emptyset, \{B\}) \\ (A \rightarrow BA, \emptyset, \{B\}), & (B \rightarrow B, \emptyset, \emptyset), \\ (A \rightarrow A, \{B\}, \emptyset), & (B \rightarrow A, \emptyset, \emptyset), \\ (A \rightarrow B, \emptyset, \emptyset). & \end{array}$$

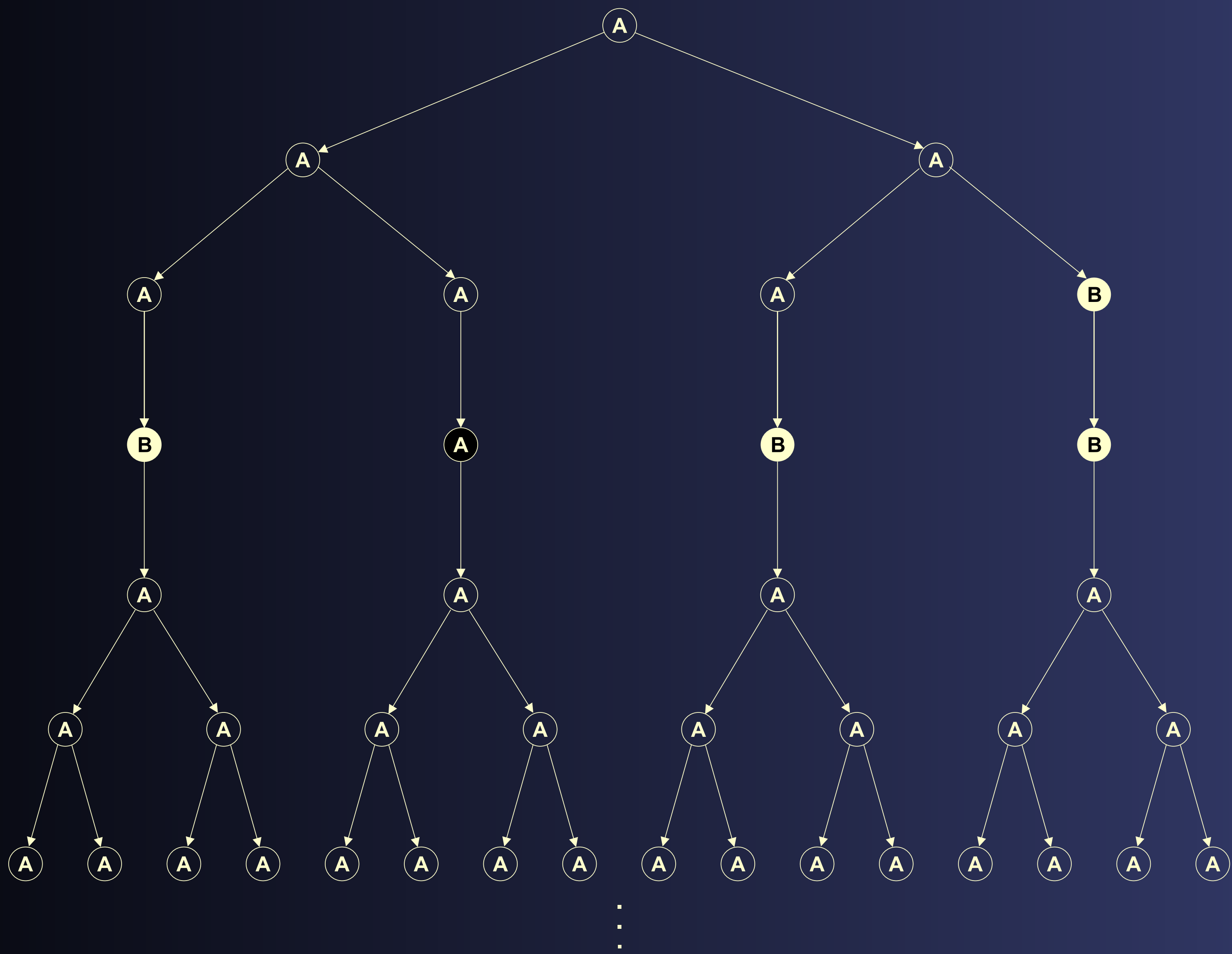
Symbols

- A – healthy cell
- B – virus-infected cell

Healthy Development



Development With a Stagnating Period



Degeneration of a Red Alga

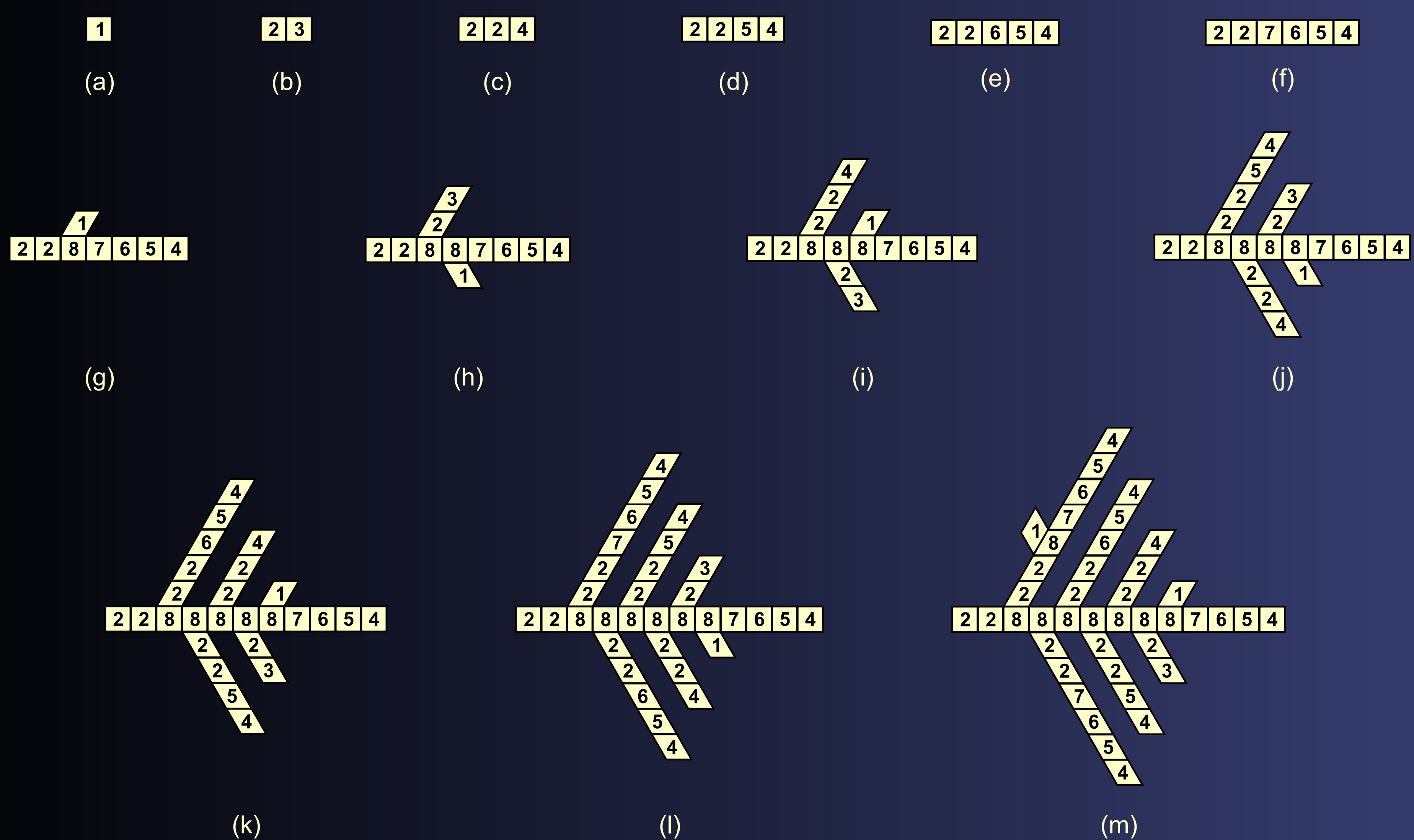
The following model describes degeneration of a red alga, where only the main stem is able to create new branches while all the other branches lengthen themselves without producing new branches.

Model

Consider a forbidding 0L grammar, G, with productions:

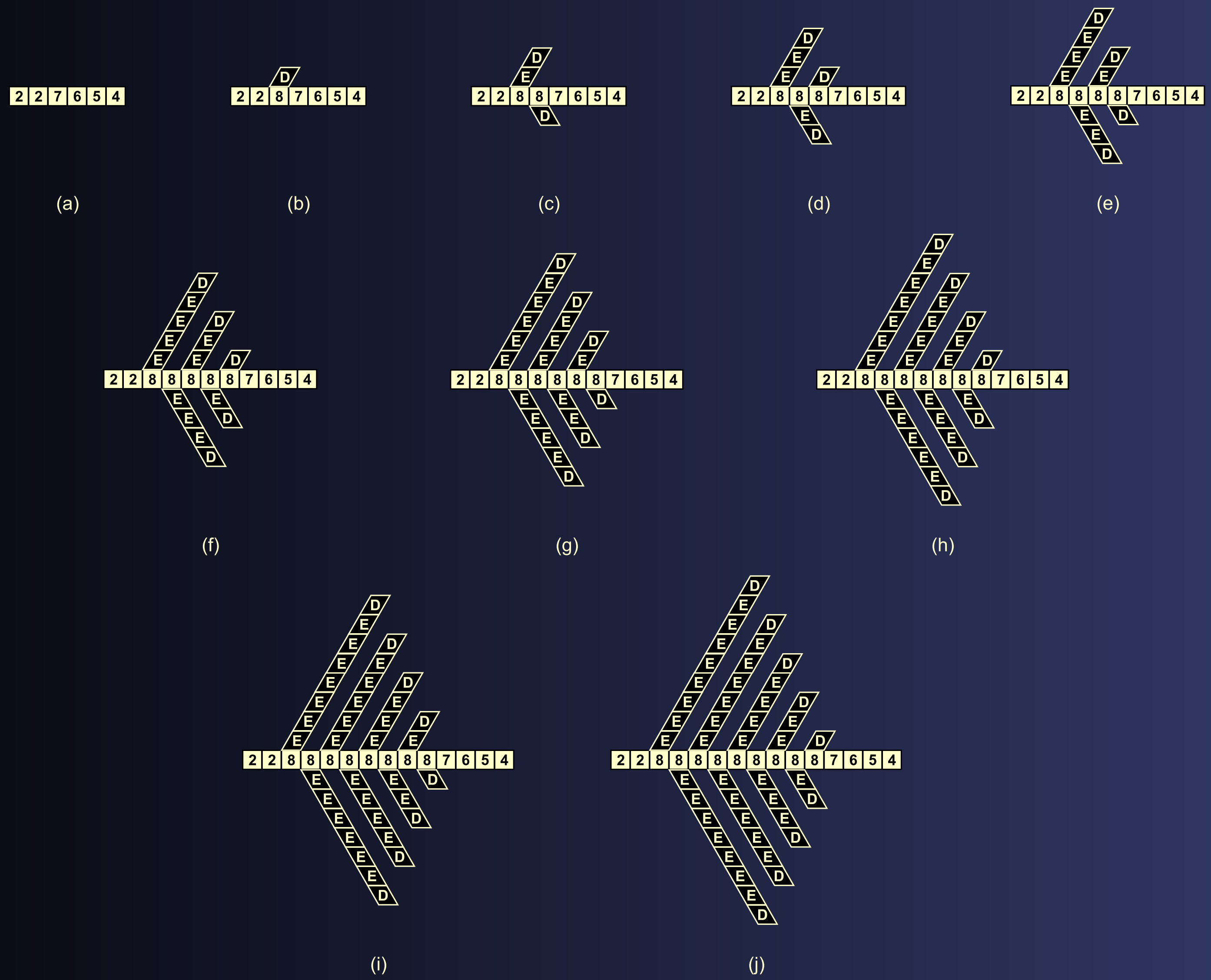
$$
\begin{array}{llll}
(1 \rightarrow 23, \emptyset), & (2 \rightarrow 2, \emptyset), & (3 \rightarrow 24, \emptyset), & (4 \rightarrow 54, \emptyset), \\
(5 \rightarrow 6, \emptyset), & (6 \rightarrow 7, \emptyset), & (7 \rightarrow 8[1], \{D\}), & (8 \rightarrow 8, \emptyset), \\
([\rightarrow [, \emptyset), & (] \rightarrow], \emptyset), & (7 \rightarrow 8[D], \emptyset), & \\
(D \rightarrow ED, \emptyset), & (E \rightarrow E, \emptyset). & &
\end{array}
$$

Healthy development



Degeneration of a Red Alga (#3)

Degeneration after application of $(7 \rightarrow 8[D], \emptyset)$.



In this case study, we extend parametric 0L grammars by permitting context conditions.

Parametric 0L Grammars operate on symbols with attached vectors of parameters.
Example of a production:

$$A(x) \ : \ x < 7 \quad \rightarrow \quad A(x+1)D(1)B(3-x)$$

This production rewrites $A(x)$ to $A(x+1)D(1)B(3-x)$ provided that $x < 7$.

Parametric 0L Grammars with Permitting Conditions introduce the concept of permitting context conditions into these grammars.
Example of a production:

$$A(x) \, ? \, B(y), C(r,z) \ : \ x < r+z \quad \rightarrow \quad D(x)E(y+r)$$

This production rewrites $A(x)$ to $D(x)E(y+r)$ provided that there is an occurrence of $B(y)$ and an occurrence of $C(r,z)$ in the rewritten sentential form and the value of the logical expression $x < r+z$ is true.

Model

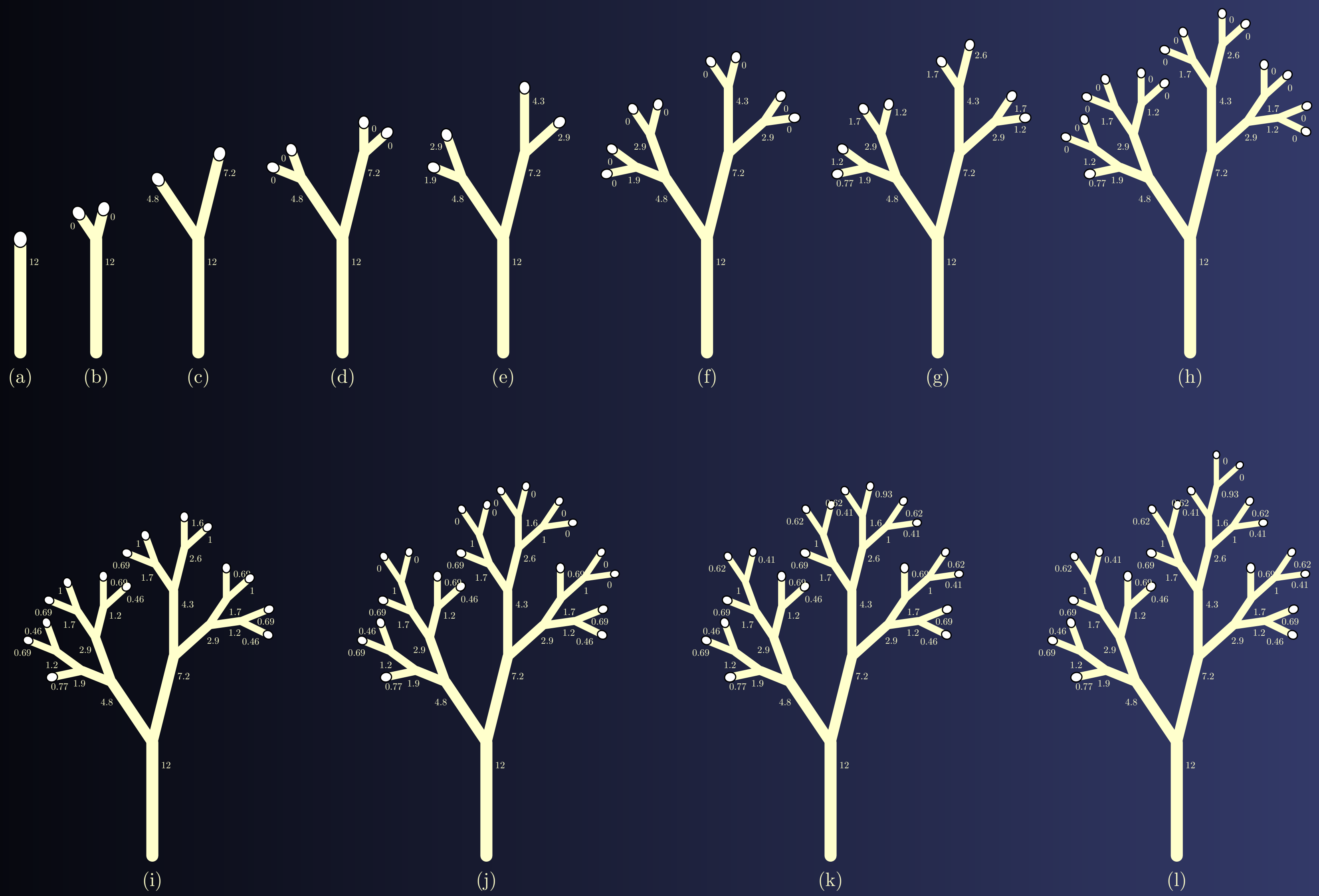
This model simulates simple resource flow distribution in a growing plant:

$$
\begin{aligned}
axiom &: \quad I(1, 1, e_{root})\, A(1) \\
p_1 &: \quad I(id, c, e)\ \textbf{?}\ I(id_p, c_p, e_p) \ : \ id_p == \lfloor id/2 \rfloor \\
& \quad \rightarrow \ I(id, c, c * e_p) \\
p_2 &: \quad A(id)\ \textbf{?}\ I(id_p, c, e) \ : \ id == id_p \text{ and } e \geq e_{th} \\
& \quad \rightarrow \ [+(\alpha)\, I(2*id+1, \gamma, 0)\, A(2*id+1)] \\
& \quad /(\gamma)\, I(2*id, 1-\gamma, 0)\, A(2*id)
\end{aligned}
$$

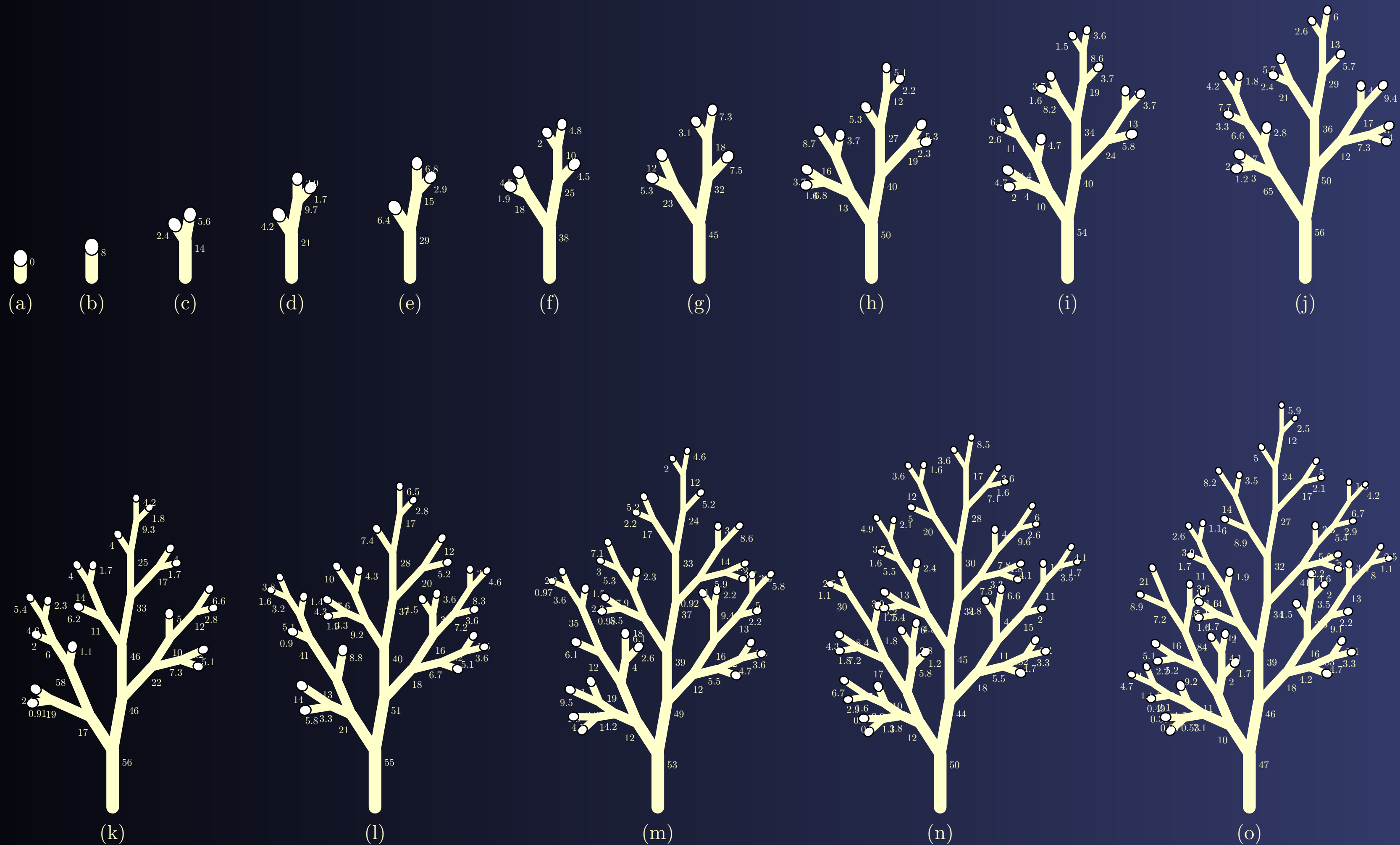
Symbols

- $I(id, c, e)$ – an internode with a unique identifier id, a flux coef. c, and a flux value e
- $A(id)$ – an apex adjacent to the internode with given identifier id
- [,] – branch delimiters
- $+(\alpha), /(\alpha)$ – rotation of branches

Developmental Stages of the Plant



A More Realistic Model Based on Context Conditions



Conclusion

Main Results

- new grammars with simple productions and easy-to-use context conditions
- new characterizations of **CS** and **RE** language families by these grammars
- reduced versions of grammars with context conditions
- sequential and parallel versions of these grammars
- formalization of the derivation similarity
- real-world applications in biology

Future Research

- investigation of another types of conditional grammars
- reduction of parallel conditional grammars
- new grammatical transformations with better derivation-simulation properties
- new applications in computer science areas, such as compilers
- new applications in other science areas, such as microbiology and genetics