CSCI 6304: Visual Languages

A Shape-Based Computational Model

November 2, 2005

Matt.Boardman@dal.ca

Research Article

- “Computing with Shapes”
  - Paulo Bottoni (Rome)
  - Giancarlo Mauri (Milan)
  - Piero Mussio (Brescia)
  - Gheorghe Păun (Bucharest)
  - *Journal of Visual Languages and Computing, Vol. 12, No. 6, 2001*

- Main concept: “Shape Completion System”

Outline

- Introduction and Terminology
- Shape Completion Systems
- Turing Computational Completeness
- Variations
- Open Issues
- Conclusions

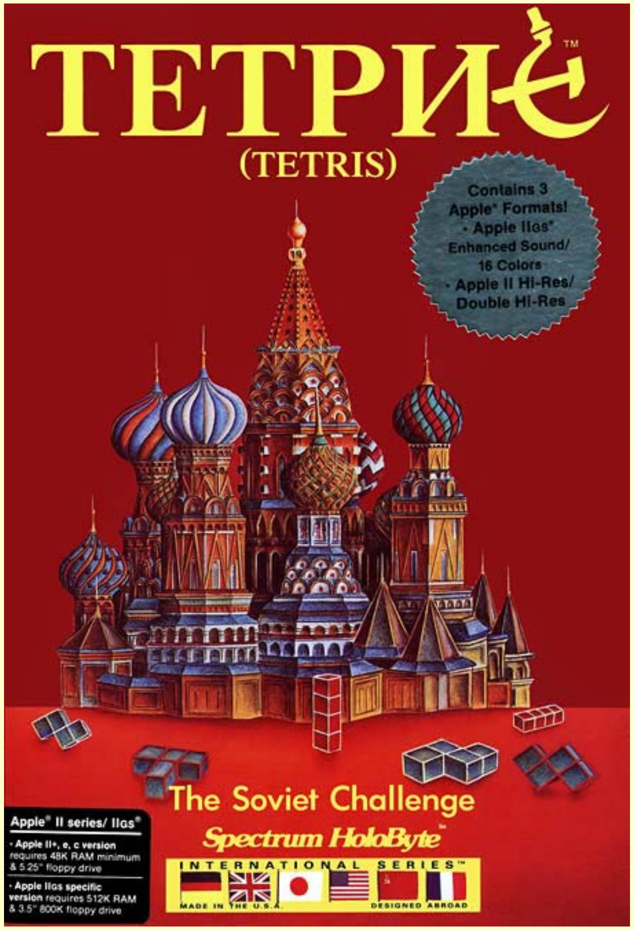
Introduction

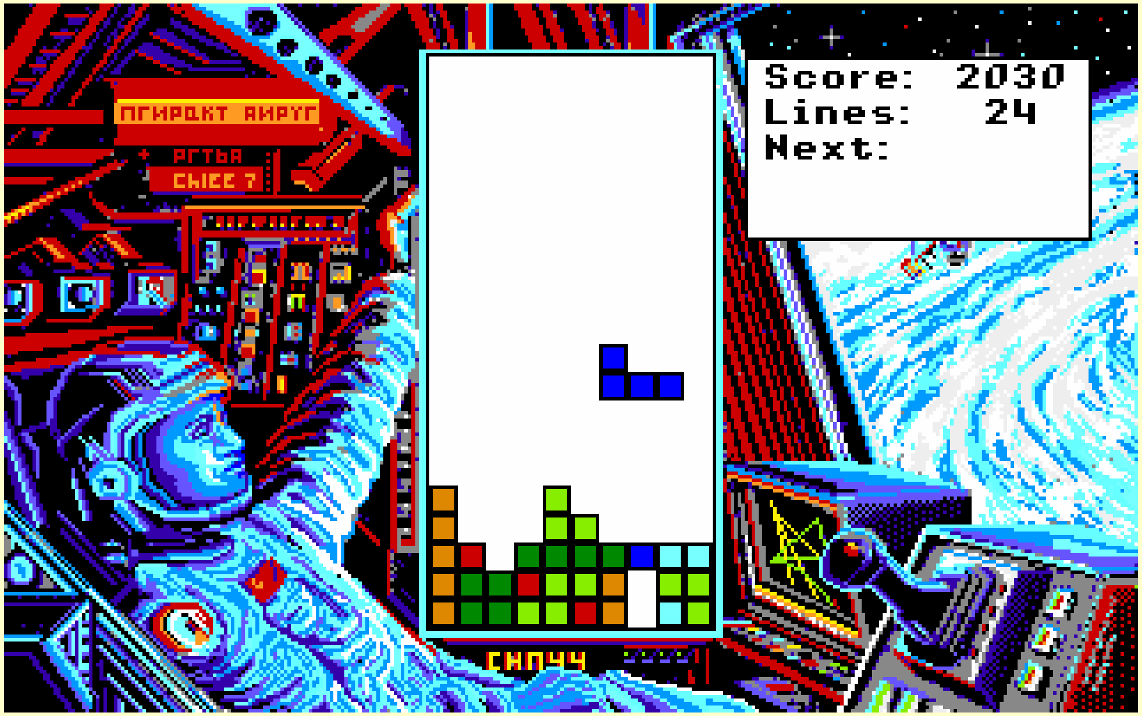
- Visual languages address communication challenges
- Computability framework: *Shape Completion Systems*
  - Associate meaning with the interaction of shapes
  - Use this interactive meaning to create *visual sentences*
  - *Interactive visual languages* (IVL) define sets of visual sentences
  - A Shape Completion System is an implementation of IVL

Introduction: Familiar Shape Completion Systems

Image: Wikipedia (Dominoes)

Introduction: Familiar Shape Completion Systems

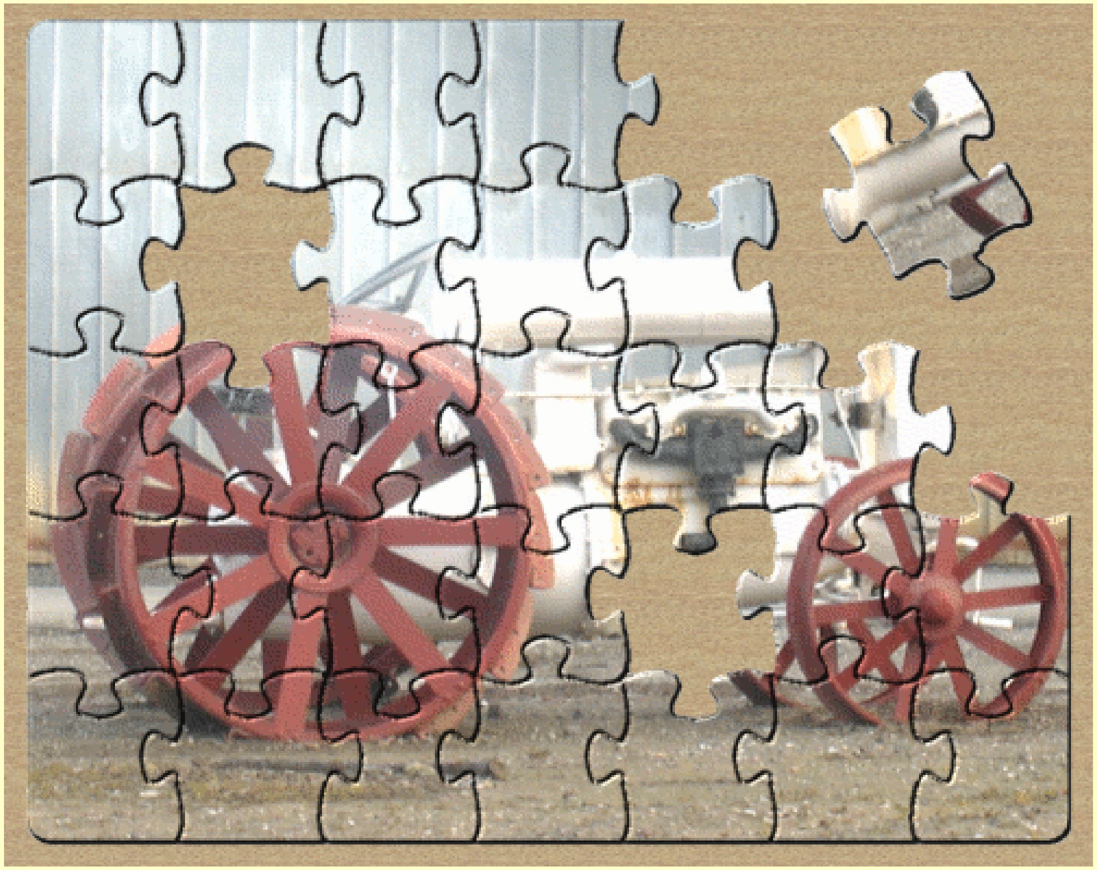




Images: Apple IIGS Gaming Memory Fairway

Introduction: Familiar Shape Completion Systems

- Shape completion systems don't need to be square:



- … but here we will only consider the square case.

Image: The GIMP Documentation (Jigsaw Filter Example)

Introduction: Applications and Related Work

- Pattern recognition and image processing
  - Algebraic characterization of images or time series
- Urban planning and industrial manufacturing
  - Shape-fitting algorithms to assemble parts in minimum space
- Plane tessellation (mosaics)
  - Based on connecting tiles with like edges (colours, symbols)
- DNA computing
  - A similar system, based on DNA replication (Kari et al, 1998)

Shape Completion Systems

- Consider a set of "polyominoes" in two-dimensional space, created from square "pixels"

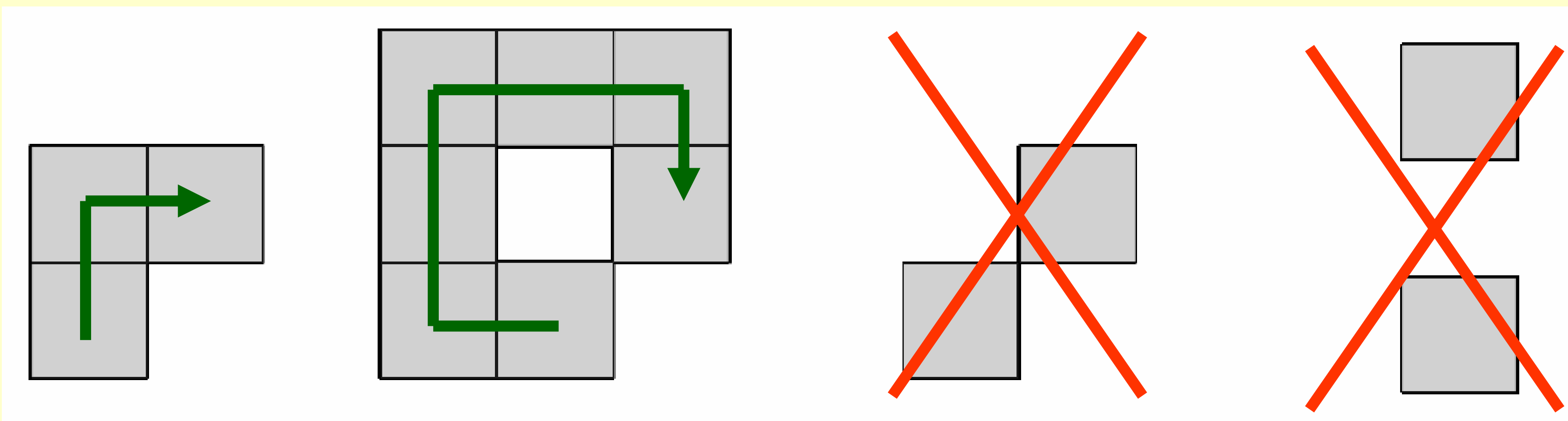


Image adapted from: Bottoni et al, 2001

Shape Completion Systems

- In formal language theory, we define:
  - an *alphabet* "V" is a finite, non-empty set of abstract symbols

    e.g. $V = \{ \mathrm{a, b, c} \}$
  - the *empty string* λ
  - a *free monoid* "V^*" is the set of possible sequences that can be created by combining zero or more elements of V

    e.g. $V^* = \{ \lambda, \mathrm{a, ab, abc, caacb}, \ldots \}$
  - a *free semigroup* "V^+" is the non-empty subset of V^*

    e.g. $V^+ = \{ \mathrm{a, ab, abc, caacb}, \ldots \}$
  - the *length* $|w|$ of a word $w \in V^*$

Shape Completion Systems

- Two-dimensional shapes map to one-dimensional sentences (i.e. a sequence of operations)
- Interactions between different polyominoes describe a meaningful "language"
- Reveal implicit and explicit dependencies
  - e.g. We cannot add an int to a float without an intermediate step
  - e.g. "I" before "E" except after "C"

Shape Completion Systems

- We can define a “correct” computation:
  1. Start with a specific polyomino
  2. Add other polyominoes in a “sequence mapping”

     A sequence mapping defines which polyominoes can be added at each step, based solely on the previous polyomino.
  3. At all steps in the computation, the adjoined polyominoes must be orthogonally connected

     Orthogonal means horizontal or vertical, but not diagonal.
  4. To be considered correct, we must end with a complete rectangle

Shape Completion Systems

- A "*Shape Completion System*":

$$\gamma = (V, P, p_0, next, lab)$$

- … where:
  - V is an alphabet
  - P is a finite set of polyominoes
  - p_0 is the initial polyomino, $p_0 \in P$
  - $next$ is a sequence mapping, $next: P \rightarrow 2^P$ (a transition mapping)
  - lab is the label mapping, $lab: P \rightarrow V^*$

Shape Completion Systems

- A "*Computation*" is a sequence:

$$\sigma = p_0 \, p_1 \, p_2 \ldots p_n \,, \quad n \geq 1$$

- … where the sequence of polyominoes is controlled by the sequence mapping:

$$p_i \in next\,(\,p_{i-1}\,)\,, \; 1 \leq i \leq n$$

- Without this condition, we would call this a "simple" shape completion system

i.e. $next\,(\,p\,) = P \quad \forall \; p \in P$

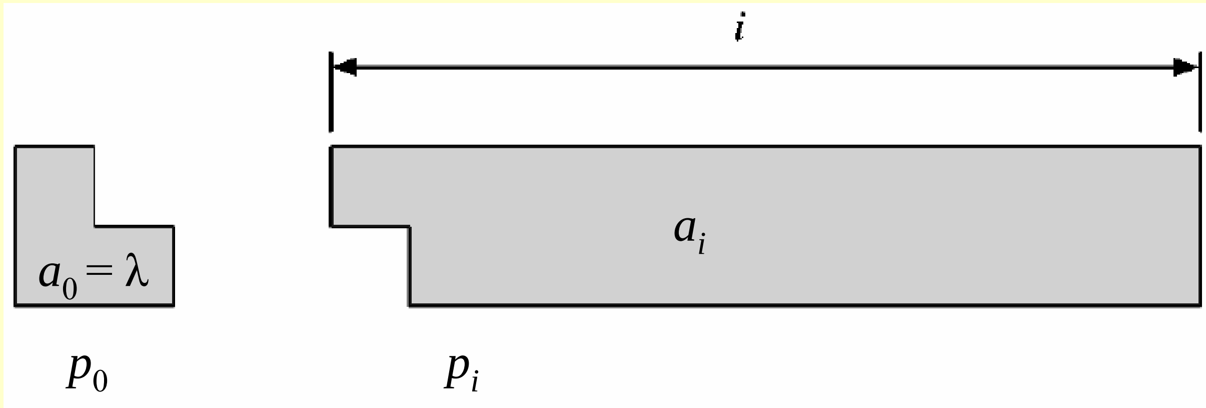
Shape Completion Systems

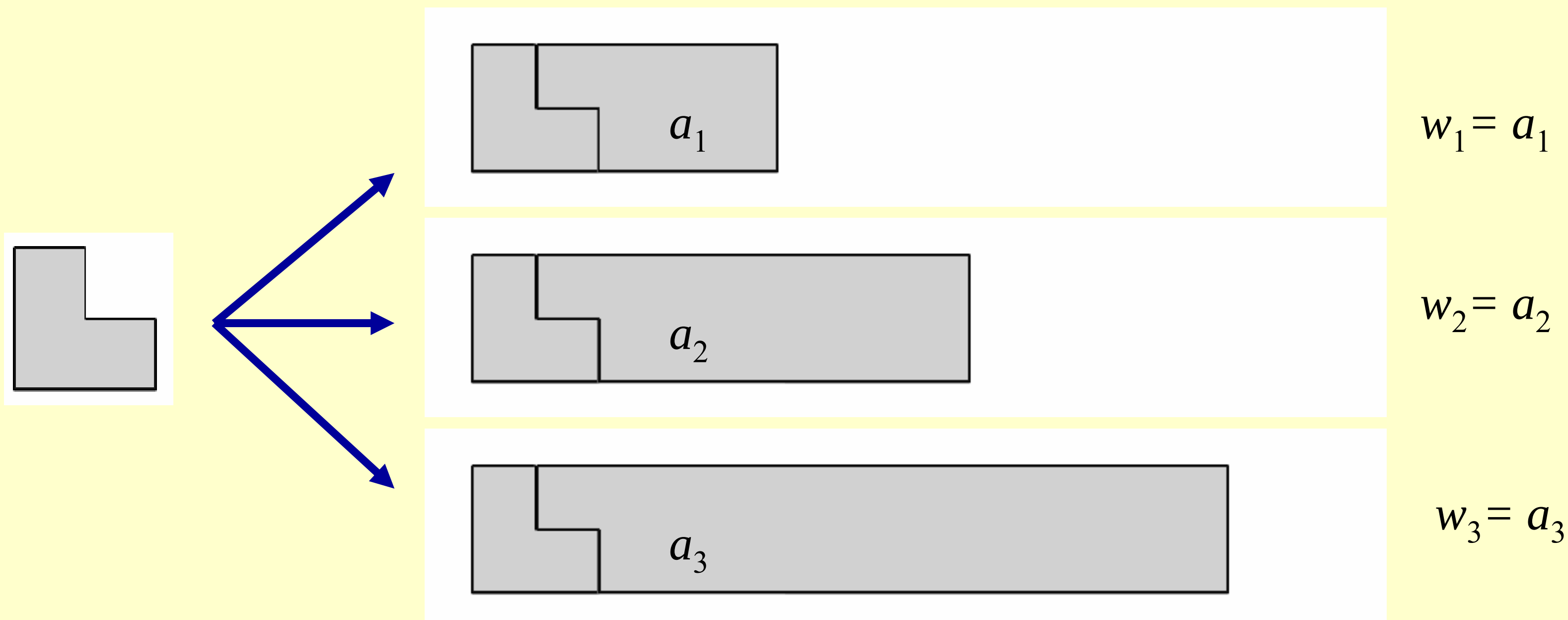
- We can use the label mapping to associate a string with each computation, simply through concatenation

  i.e. $lab(\sigma) = lab(p_0)lab(p_1)lab(p_2)\ldots lab(p_n)\,,\qquad n \geq 1$

- The resulting language of all strings that can be generated from the computations in γ is denoted $L(\gamma)$
- A sparse definition of lab with values in $V \cup \{\lambda\}$ indicates that γ is "reduced"
- Authors' assertion: *"Any finite language can be generated by a reduced, simple shape completion system."*

Shape Completion Systems: Example 1





• • •

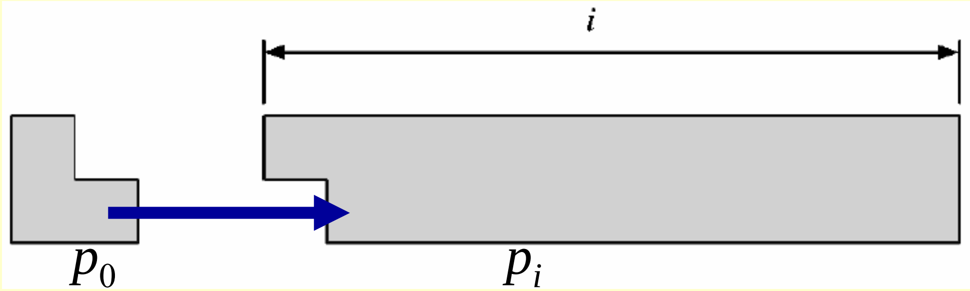
Image adapted from: Bottoni et al, 2001

Shape Completion Systems: Example 1

$V = \{ a_1, \ldots, a_n \}$

$L_1 \subseteq V^*$

$L_1 = \{ w_1, \ldots, w_m \}$



For each w_i where $i = 1, 2, \ldots, m$

we consider the set of polyominoes

$$P = \{ p_0 , p_i \}$$

to create:

$$\gamma_1 = (V, P, p_0, lab)$$

where $lab(p_0) = \lambda$

$lab(p_i) = a_i$

Result: $L(\gamma_1) = L_1$

(Simple, but not necessarily reduced)

Image adapted from: Bottoni et al, 2001

Shape Completion Systems: Example 2

$V = \{ a_1, \ldots, a_n \}$

$L_2 \subseteq V^*$

$L_2 = \{ w_1, \ldots, w_m \}$

For each $w_i = a_{i,1}\, a_{i,2} \ldots a_{i,k_i}$

where $\quad a_{i,j} \in V \qquad 1 \leq j \leq k_i$

$\qquad\qquad\qquad\qquad 1 \leq i \leq m$

we consider polyominoes

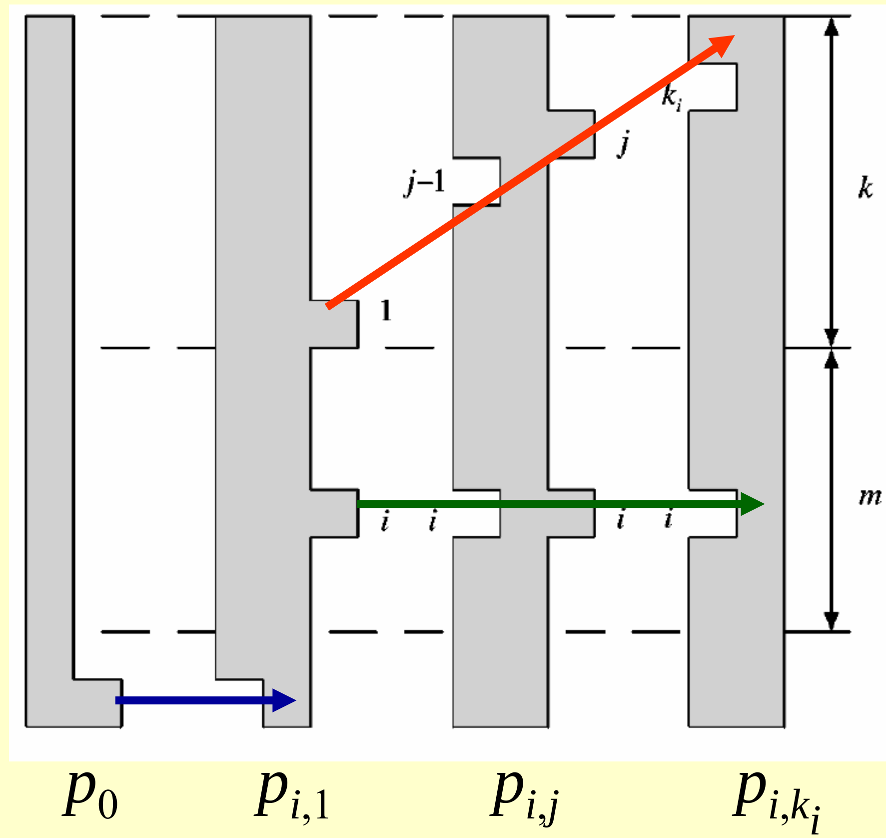
$$P = \{ p_0 , p_{i,j} \}$$

to create:

$$\gamma_2 = (V, P, p_0, lab)$$

where $\quad lab(p_0) = \lambda$

$\qquad\qquad lab(p_{i,j}) = a_{i,j}$



Result: $\quad L(\gamma_2) = L_2$

Image adapted from: Bottoni et al, 2001

Turing Computational Completeness

- What does it mean to be Turing-complete?
  - System can be emulated by a Turing machine
  - Imperative languages: BASIC, C
  - Object-oriented languages: C++, Java, Smalltalk
  - Visual languages: Prograph, LabView
  - But not: SQL, Spreadsheets

Turing Computational Completeness

- Recursively Enumerable languages can be evaluated by a Turing Machine
- "Pure characterization" of shape-based languages in this paper is formal proof that IVL have the same computational power as Turing machines!
  - no non-pictorial elements
  - no non-terminal symbol or operation
  - no matching colours
  - no deformations of placed elements

Turing Computational Completeness

- Chomsky Hierarchy:
  - Noam Chomsky based a hierarchy of grammars on linguistics
  - *Type 0*: Recursively Enumerable Grammars
    - May be evaluated using Turing machine
  - *Type 1*: Context Sensitive Grammars
    - May be evaluated using Linear-Bounded Automaton
  - *Type* 2: Context Free Grammars
    - May be evaluated using Pushdown automaton
  - *Type 3*: Regular Grammars
    - May be evaluated using Finite State Machine

Image in public domain

Turing Computational Completeness

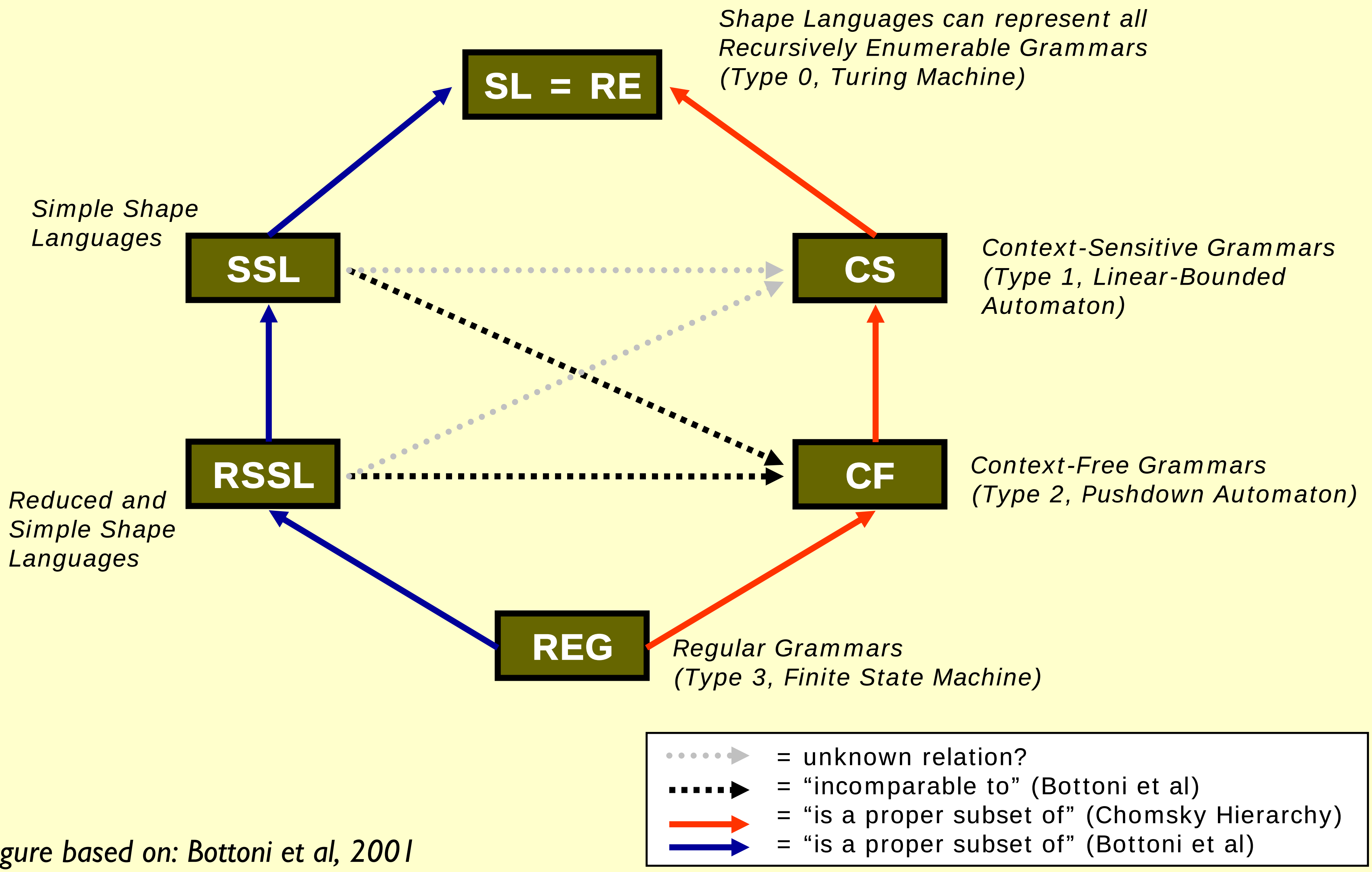


Figure based on: Bottoni et al, 2001

Additional Variations

- Some stronger variations:
  - Specify *stop* polyomino, in addition to initial polyomino
  - Impose a limit on the *height* of final rectangle
  - Matching *colours* or *symbols* (Wang, Penrose tiles)
  - Designate particular segments as *sticky* (Kari DNA)
  - *Stationary* computations: stop when no steps possible

- Proposed computational complexity measure:
  - Ratio of surface area of final rectangle to length of output string

Open Issues

- Implementation issues:
  - "Shape fitting" algorithm?
    - See e.g. S. Har-Peled, Y. Wang, "Shape Fitting with Outliers," *SIAM Journal of Computing*, 33(2), 2004, pp. 269-285.
  - Allow rotation, other transformations?
    - Left to particular implementation (not allowed here).
  - Design of *next* sequence mapping?
    - Left to particular implementation (none specified here).
  - How to address logical decidability?
    - There may exist formulas or inputs which are *undecidable*, i.e. there is no algorithm with a finite number of steps to determine semantic validity.
- Advantages? Disadvantages?
    - To compare a similar system with biological plausibility, see L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa, S. Yu, "DNA computing, sticker systems, and universality", *Acta Informatica*, 35(6), pp. 401-420, 1998.

Conclusions

- Complex languages can be created using Shape Completion Systems
  - Even without *next* sequence mapping (i.e. SSL or RSSL)
- IVL are Turing-complete
  - Shape Completion System is a variant of IVL
- "A picture is worth 1000 words"
  - Mathematical proof of the power and complexity of visual languages

References

- P. Bottoni, G. Mauri, P. Mussio, Gh. Păun, "Computing with Shapes," *Journal of Visual Languages and Computing,* 12(6), 2001, pp. 601-626.
- P. Bottoni, M. F. Costabile, S. Levialdi, P. Mussio, "Defining Visual Languages for Interactive Computing," *IEEE Transactions on Systems, Man, and Cybernetics – Part A,* 27(6), 1997, pp. 773-782.
- The GIMP Documentation, "Jigsaw Filter Example" (Figure 10.40), [http://docs.gimp.org].
- S. Har-Peled, Y. Wang, "Shape Fitting with Outliers," *SIAM Journal of Computing*, 33(2), 2004, pp. 269-285.
- L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa, S. Yu, "DNA computing, sticker systems, and universality", *Acta Informatica*, 35(6), pp. 401-420, 1998.
- A. Pajitnov, "ТЕТРИС (Tetris): The Soviet Challenge," *Academy Soft – ELORG / Spectrum Holobyte – Sphere Inc.*, 1987. Images: A. Lee, "The Apple IIGS Gaming Memory Fairway", 2001 [http://www.whatisthe2gs.apple2.org.za].
- Schadel, "Image of Turing Machine" (in public domain), 2005.
- M. A. Tapia, J. P. Duarte, "Shape Grammars," *MIT*, 1999 [http://shapegrammar.org].
- Wikipedia contributors, "Dominoes" (Image), "Chomsky Hierarchy," "Decidability (logic)," "Free Monoid," "Recursively Enumerable Language," "Turing Completeness," "Turing Machine," *Wikipedia: The Free Encyclopedia*, 2005, [http://en.wikipedia.org/wiki].

Appendix: Visual Sentences

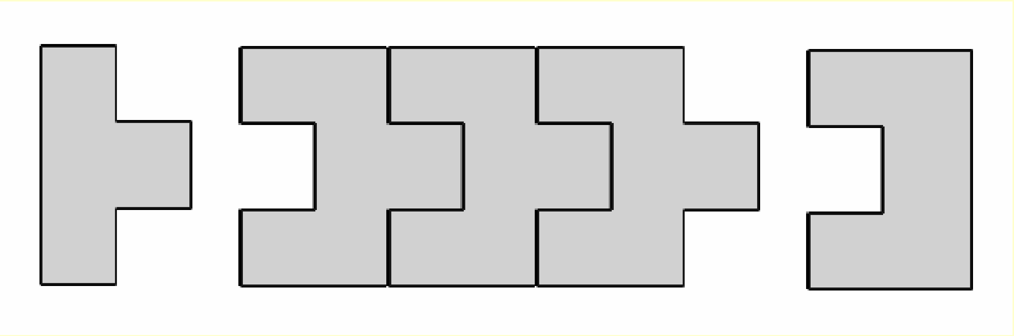
- For Interactive Visual Languages (IVL), we define:
  - "i" *image:* an abstract arrangement of shapes
  - "d" *description:* the meaning of shape interactions
  - "int" *interpretation:* explicit relation from $i \rightarrow d$
  - "mat" *materialization:* explicit relation from $d \rightarrow i$
  - "vs" *visual sentence:* quadruple of these four definitions

i.e. $vs = (i, d, int, mat)$

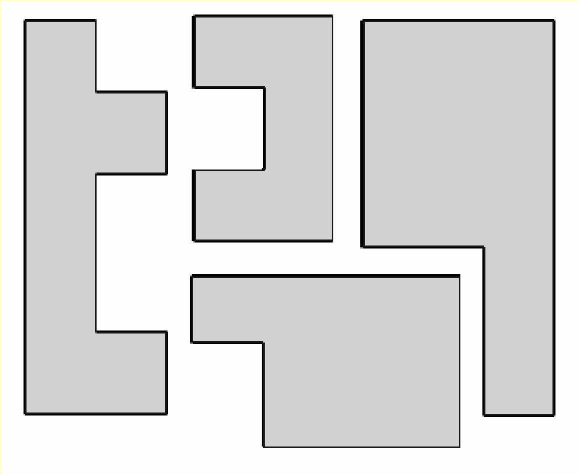
Source: Bottini et al, 1997

Appendix: Turing Computational Completeness

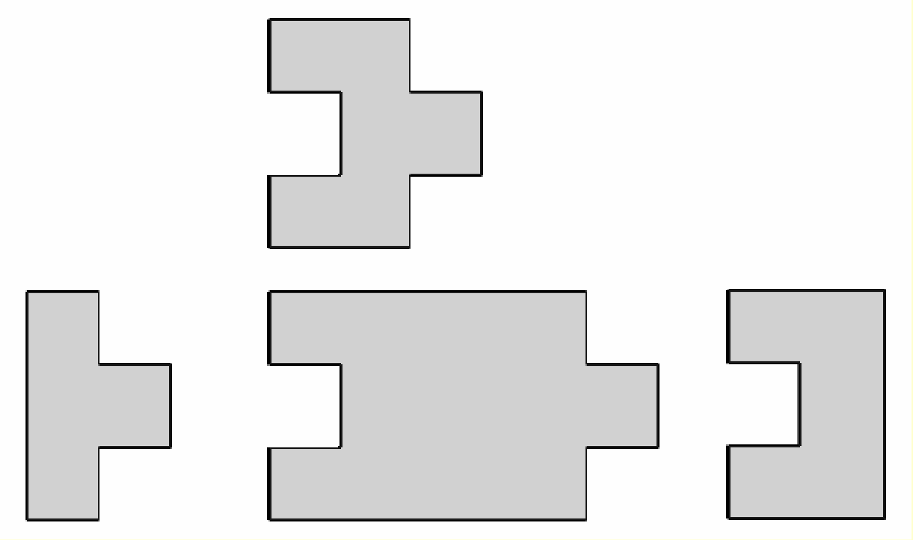
- Looping constructs:



- Conditional statements, Boolean operations:



"AND"



"OR"

Appendix: Proof of Theorem 5

- **Theorem**: Reduced, simple shape completion systems can produce fairly complex languages which are non-context-free.

  $$RSSL - CF \neq \emptyset$$

Appendix: Proof of Theorem 5

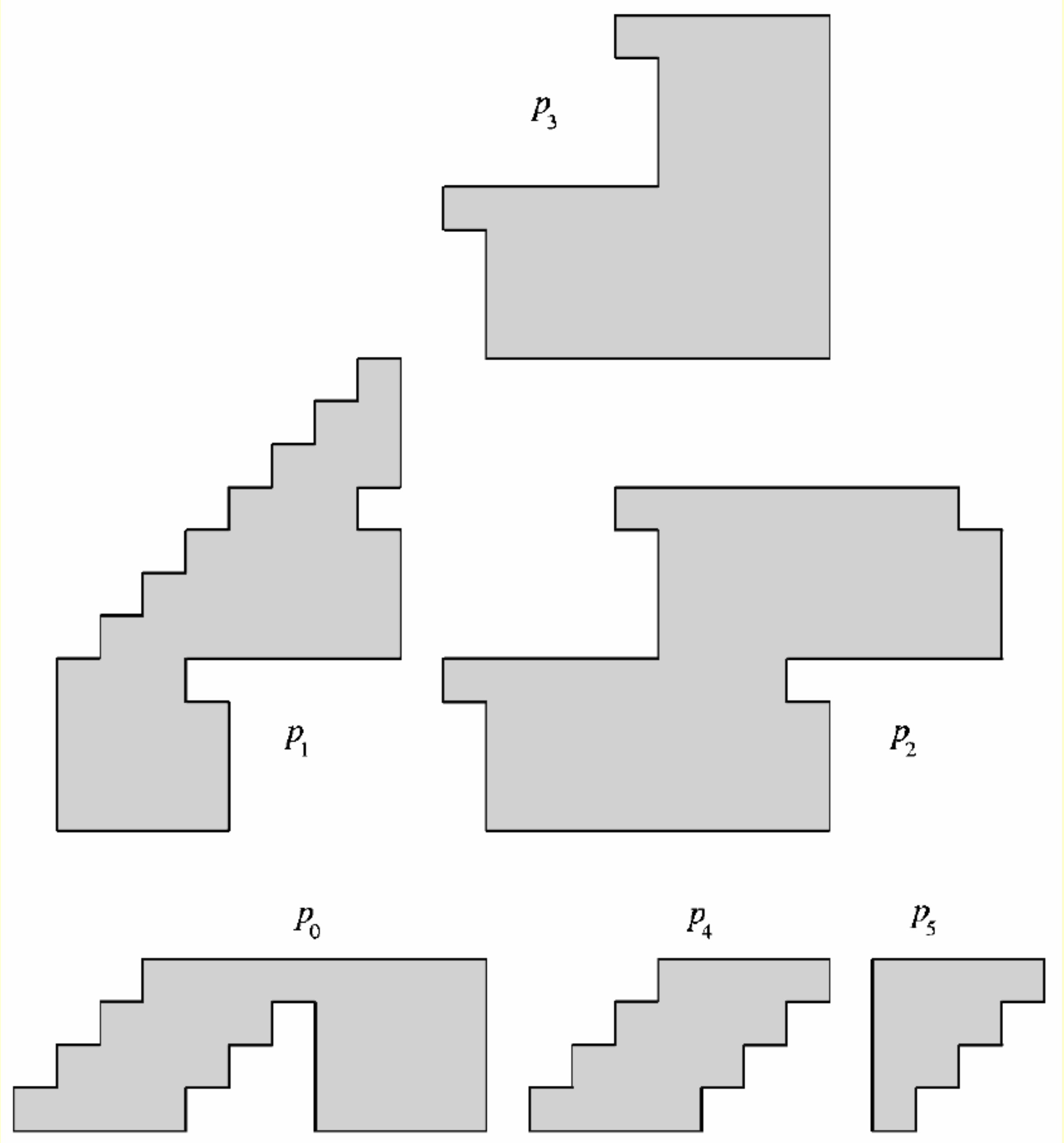
Here we consider a one-letter, non-regular, simple, reduced shape completion system:

$\gamma = (\ \{\ a\ \}, P, p_0, lab\ )$

where:

$P = \{\ p_0, p_1, p_2, p_3, p_4, p_5\ \}$

$lab(\ p_i\ ) = a \qquad 0 \le i \le 5$

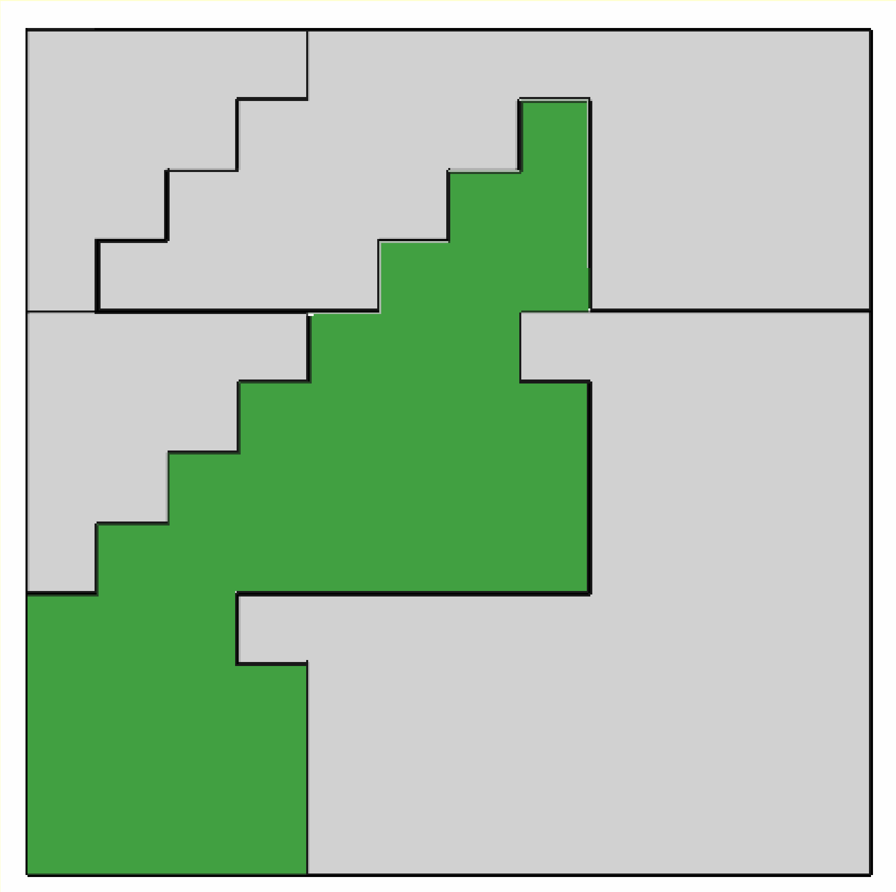


Appendix: Proof of Theorem 5

- These shapes can only be used to create squares (no rectangles are possible).

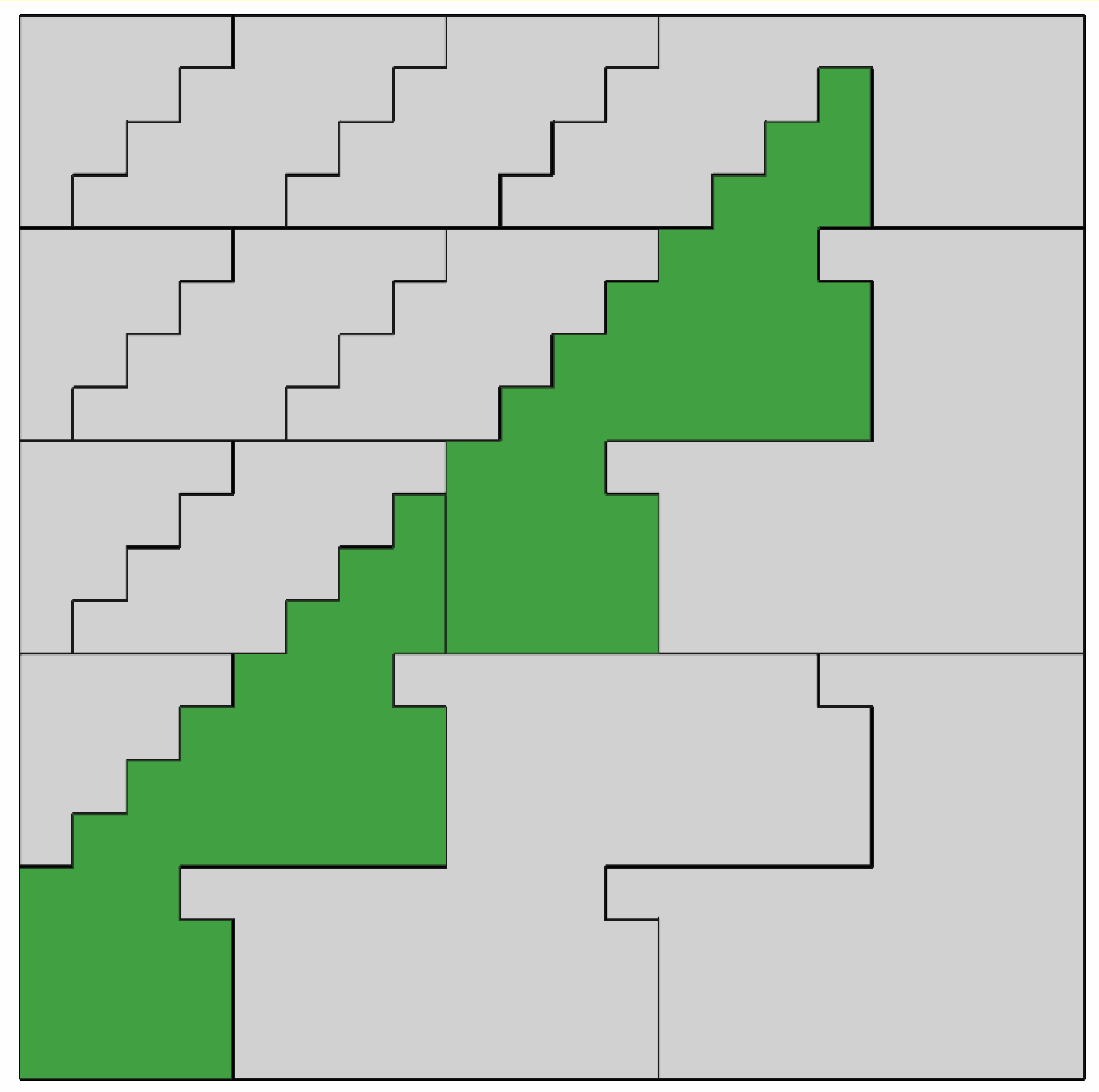
Call n the number of times that p_1 is used (shown green).

For example, here we have $n = 1$.



Appendix: Proof of Theorem 5

Here we have $n = 2$.



Appendix: Proof of Theorem 5

- We can calculate the size of the resulting square, as a function of n:

$$S(n) = (\ 8n + 4\ )^2 \qquad\qquad n \leq 1$$

- In constructing a square of size n, we use:

one copy of p_0

n copies of of p_1 $\qquad$ $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ copies of of p_2

n copies of of p_3 $\qquad$ $2(n\text{-}1)+2 = 2n$ copies of of p_5

$$\sum_{i=1}^{2(n-1)} i + 2(n-1) = 2n^2 - n - 1 \text{ copies of of } p_4$$

Appendix: Proof of Theorem 5

- Since all polyominoes are labelled with a, we get a string of a's for a square of size n:

$$\begin{aligned}|w_n| &= 1+n+n+\frac{n(n-1)}{2}+2n+2n^2-n-1\\ &= \frac{5n(n+1)}{2}\end{aligned}$$

… which is always an integer, since $n(n+1)$ is always even.

Appendix: Proof of Theorem 5

- We therefore have:

$$L(\gamma) = \left\{ a^{[5n(n+1)/2]} \middle| n \geq 1 \right\}$$

… which is not a regular language. Since one-letter, context-free languages are regular, $L(\gamma_2) \notin CF$ (i.e. it is non-context-free).

QED