Language-Processing Problems

Roland Backhouse

DIMACS, 8th July, 2003

Introduction

"Factors" and the "factor matrix" were introduced by Conway (1971).

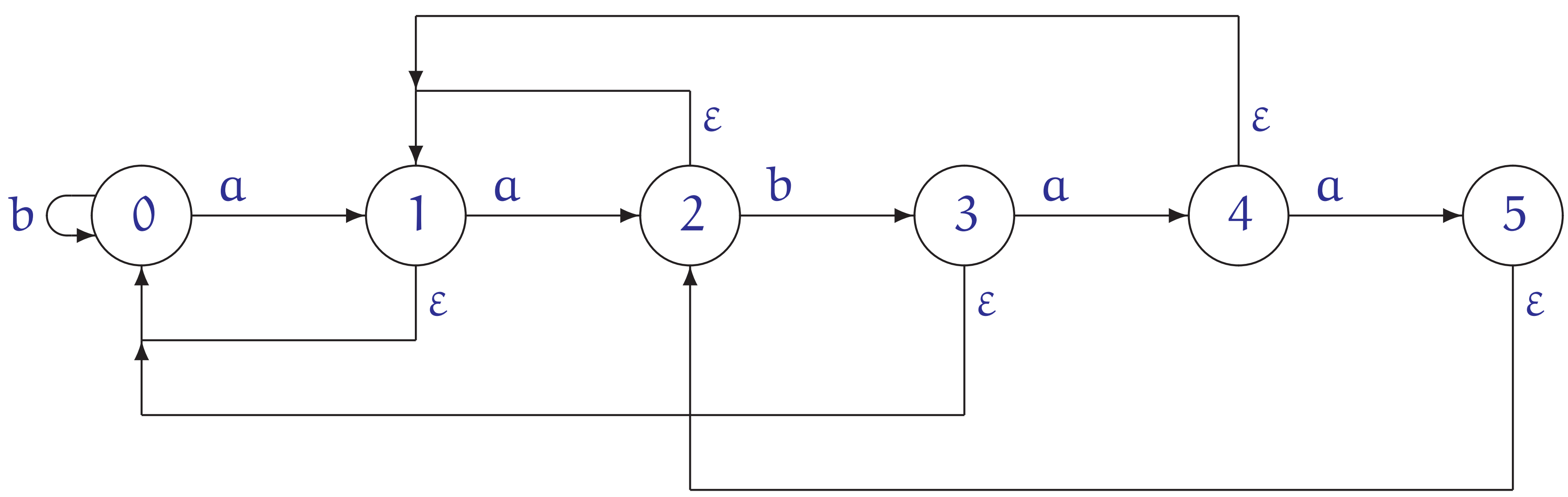
He used them very effectively in, for example, constructing biregulators.

Conway's discussion is wordy, making it difficult to understand. There are also occasional errors which are difficult to detect and add to the confusion. ("The theorem does prevent E from occurring twice" should read "The theorem does *not* prevent E from occurring twice.")

KMP Failure Function (pattern $aabaa$)

node i	1	2	3	4	5
failure node $f(i)$	0	1	0	1	2

Factor Graph (language $\Sigma^* aabaa$)



Language Problems

$$S \ ::= \ aSS \mid \varepsilon \ .$$

Is-empty

$$S = \phi \quad \equiv \quad (\{a\} = \phi \vee S = \phi \vee S = \phi) \ \wedge \ \{\varepsilon\} = \phi \ .$$

Nullable

$$\varepsilon \in S \quad \equiv \quad (\varepsilon \in \{a\} \wedge \varepsilon \in S \wedge \varepsilon \in S) \ \vee \ \varepsilon \in \{\varepsilon\} \ .$$

Shortest word length

$$\#S \ = \ (\#a + \#S + \#S) \downarrow \#\varepsilon \ .$$

Non-Example

$$aa \in S \quad \not\equiv \quad (aa \in \{a\} \ \wedge \ aa \in S \ \wedge \ aa \in S) \ \vee \ aa \in \{\varepsilon\} \ .$$

Fusion

Many problems are expressed in the form

$$\mathit{evaluate} \circ \mathit{generate}$$

where *generate* generates a (possibly infinite) candidate set of solutions, and *evaluate* selects a best solution.

Examples:

$$\mathit{shortest} \circ \mathit{path} \quad ,$$

$$(x\in) \circ \mathsf{L} \quad .$$

Solution method is to *fuse* the generation and evaluation processes, eliminating the need to generate all candidate solutions.

Conditions for Fusion

Fusion is made possible when

- *evaluate* is an adjoint in a *Galois connection*,
- *generate* is expressed as a *fixed point*.

Algorithms for solving resulting fixed point equation include

- brute-force iteration,
- Knuth's generalisation of Dijkstra's shortest path algorithm. .

Solution method typically involves *generalising* the problem.

Galois Connections

Suppose $\mathcal{A} = (A, \sqsubseteq)$ and $\mathcal{B} = (B, \preceq)$ are partially ordered sets and suppose $F \in A \leftarrow B$ and $G \in B \leftarrow A$. Then (F, G) *is a Galois connection of* $\mathcal{A}$ *and* $\mathcal{B}$ iff, for all $x \in B$ and $y \in A$,

$$F(x) \sqsubseteq y \;\equiv\; x \preceq G(y) \quad .$$

Examples

Negation:

$$\neg p \Rightarrow q \;\equiv\; p \Leftarrow \neg q \quad .$$

Ceiling function:

$$\lceil x \rceil \leq n \;\equiv\; x \leq n \quad .$$

Maximum:

$$x \uparrow y \leq z \;\equiv\; x \leq z \wedge y \leq z \quad .$$

Even (divisible by two):

$$\textsf{if}\ b \rightarrow 2 \ \square\ \neg b \rightarrow 1\ \textsf{fi} \setminus m \;\equiv\; b \Rightarrow \textsf{even}(m) \quad .$$

Parsing

$$x{\in}S \Rightarrow b \quad \equiv \quad S \subseteq \mathsf{if}\ b \rightarrow \Sigma^* \ \Box \ \neg b \rightarrow \Sigma^* - \{x\}\ \mathsf{fi} \quad .$$

Shortest Word (Path)

Let $\Sigma^{\geq k}$ denote the set of all words over alphabet Σ of length at least k.

Let $\#S$ denote the length of a shortest word in the language S.

$$\#S \geq k \ \equiv \ S \subseteq \Sigma^{\geq k} \quad .$$

(Most common application is when S is the set of paths from one node to another in a graph.)

Fusion Theorem

$$F(\mu_{\preceq} g) \;=\; \mu_{\sqsubseteq} h$$

provided that

- F is a lower adjoint in a Galois connection of $\sqsubseteq$ and $\preceq$ (see brief summary of definition below)
- $F \circ g \;=\; h \circ F$.

Galois Connection

$$F(x) \sqsubseteq y \;\equiv\; x \preceq G(y) \quad .$$

F is called the *lower* adjoint and G the *upper* adjoint.

Language Recognition

Problem: For given word x and grammar G, determine $x \in L(G)$.
That is, implement

$$(x\in) \circ L \quad .$$

Language $L(G)$ is the least fixed point (with respect to the subset relation) of a monotonic function.

$(x\in)$ is the lower adjoint in a Galois connection of languages (ordered by the subset relation) and booleans (ordered by implication).
(Recall,

$$x\in S \Rightarrow b \quad \equiv \quad S \subseteq \text{if } b \rightarrow \Sigma^* \;\square\; \neg b \rightarrow \Sigma^* - \{x\} \text{ fi} \quad .)$$

Nullable Languages

Problem: For given grammar G, determine $\varepsilon \in L(G)$.

$$(\varepsilon\in) \circ L$$

Solution: Easily expressed as a fixed point computation.

Works because:

- The function $(x\in)$ is a lower adjoint in a Galois connection (for all x, but in particular for $x = \varepsilon$).
- For all languages S and T,

$$\varepsilon \in S{\cdot}T \quad \equiv \quad \varepsilon \in S \wedge \varepsilon \in T \quad .$$

Problem Generalisation

Problem: For given grammar G, determine whether all words in $\mathsf{L(G)}$ have even length. I.e. implement

$$\mathsf{alleven} \circ \mathsf{L} \ .$$

The function $\mathsf{alleven}$ is a lower adjoint in a Galois connection. Specifically, for all languages S and T,

$$\mathsf{alleven}(\mathsf{S}) \Leftarrow \mathsf{b} \quad \equiv \quad \mathsf{S} \subseteq \mathsf{if}\ \neg\mathsf{b} \rightarrow \Sigma^* \ \Box\ \mathsf{b} \rightarrow (\Sigma{\cdot}\Sigma)^*\ \mathsf{fi} \ .$$

Nevertheless, fusion *doesn't* work (directly) because

- there is no $\otimes$ such that, for all languages S and T,

$$\mathsf{alleven}(\mathsf{S}{\cdot}\mathsf{T}) \quad \equiv \quad \mathsf{alleven}(\mathsf{S}) \otimes \mathsf{alleven}(\mathsf{T}) \ .$$

Solution: Generalise by tupling: compute simultaneously $\mathsf{alleven}$ and allodd.

General Context-Free Parsing

Problem: For given grammar G, determine $x \in L(G)$.

$$(x\in) \circ L \ .$$

Not (in general) expressible as a fixed point computation.

Fusion *fails* because: for all x, $x \neq \varepsilon$, there is no $\otimes$ such that, for all languages S and T,

$$x \in S{\cdot}T \quad \equiv \quad (x \in S) \otimes (x \in T) \ .$$

CYK: Let $F(S)$ denote the relation $\langle i, j{::}\ x[i..j) \in S \rangle$.

Works because:

- The function F is a lower adjoint.
- For all languages S and T,

  $$F(S{\cdot}T) \ = \ F(S) \bullet F(T)$$

  where $B \bullet C$ denotes the composition of relations B and C.

Language Inclusion

Problem: For fixed (regular) language E and varying S, determine

$$\mathsf{S} \subseteq \mathsf{E} \quad .$$

Example: Emptiness test:

$$\mathsf{S} \subseteq \phi \quad .$$

Example: Pattern Matching: given pattern P, for each prefix t of text T, evaluate:

$$\{\mathsf{t}\} \subseteq \Sigma^* \cdot \{\mathsf{P}\} \quad .$$

Example: All words are of even length:

$$\mathsf{S} \subseteq (\Sigma \cdot \Sigma)^* \quad .$$

Language Inclusion

Problem: For fixed (regular) language E and varying S, determine

$$\mathsf{S} \subseteq \mathsf{E} \ .$$

- Function $(\subseteq \mathsf{E})$ is a lower adjoint. Specifically,

$$\mathsf{S} \subseteq \mathsf{E} \Leftarrow \mathsf{b} \quad \equiv \quad \mathsf{S} \subseteq \mathbf{if}\ \mathsf{b} \longrightarrow \mathsf{E} \ \Box\ \neg \mathsf{b} \longrightarrow \Sigma^* \ \mathbf{fi} \ .$$

- But, for $\mathsf{E} \neq \phi$ and $\mathsf{E} \neq \Sigma^*$, there is no $\otimes$ such that, for all languages S and T,

$$\mathsf{S} \cdot \mathsf{T} \subseteq \mathsf{E} \quad \equiv \quad (\mathsf{S} \subseteq \mathsf{E}) \otimes (\mathsf{T} \subseteq \mathsf{E}) \ .$$

Solution (Oege de Moor): Use factor theory to derive generalisation.

Factors

For all languages S, T and U,

$$S{\cdot}T \subseteq U \;\; \equiv \;\; T \subseteq S\backslash U \quad ,$$

$$S{\cdot}T \subseteq U \;\; \equiv \;\; S \subseteq U/T \quad .$$

Note:

$$S\backslash(U/T) \; = \; (S\backslash U)/T \quad .$$

Hence, write

$$S\backslash U/T \quad .$$

Left and Right Factors

Define the functions $\triangleleft$ and $\triangleright$ by

$$\begin{aligned} X\triangleleft &= E/X \ , \\ X\triangleright &= X\backslash E \ . \end{aligned}$$

By definition, the range of $\triangleleft$ is the set of *left* factors of E and the range of $\triangleright$ is the set of *right* factors of E.

We also have the Galois connection:

$$X \subseteq Y\triangleleft \ \equiv \ Y \subseteq X\triangleright \ .$$

Hence,

$$\begin{aligned} X\triangleleft\triangleright\triangleleft &= X\triangleleft \ , \\ X\triangleright\triangleleft\triangleright &= X\triangleright \ , \\ E\triangleleft\triangleright &= E = E\triangleright\triangleleft \ . \end{aligned}$$

The Factor Matrix

Let $\mathcal{L}$ denote the set of left factors of E.

Define the *factor matrix* of E to be the binary operator $\backslash$ restricted to $\mathcal{L}\times\mathcal{L}$. Thus entries in the matrix take the form $\mathsf{L}_0\backslash\mathsf{L}_1$ where L_0 and L_1 are left factors of E.

The factor matrix of E is denoted by $[\![\mathsf{E}]\!]$. It is a reflexive, transitive matrix.

$$[\![\mathsf{E}]\!] \quad = \quad [\![\mathsf{E}]\!]^* \quad .$$

The row and column containing individual factors, the left factors, the right factors, and E itself, is given by:

$$\begin{aligned}
\mathsf{U}\backslash\mathsf{E}/\mathsf{V} &= \mathsf{U}{\rhd\lhd}\backslash\mathsf{V}{\lhd} \quad , \\
\mathsf{V}{\lhd} &= \mathsf{E}{\lhd}\backslash\mathsf{V}{\lhd} \quad , \\
\mathsf{U}{\rhd} &= \mathsf{U}{\rhd\lhd}\backslash\mathsf{E}{\rhd\lhd} \quad , \\
\mathsf{E} &= \mathsf{E}{\lhd}\backslash\mathsf{E}{\rhd\lhd} \quad .
\end{aligned}$$

Using the Factor Matrix

Problem: For fixed regular language E and varying S, determine

$$\mathsf{S} \subseteq \mathsf{E} \quad .$$

Generalisation: For fixed regular language E and varying S, determine the relation

$$\mathsf{S} \subseteq [\![\mathsf{E}]\!] \quad .$$

(Formally, the relation $\langle \mathsf{L},\mathsf{M}{::}\ \ \mathsf{S} \subseteq \mathsf{L}\backslash\mathsf{M}\rangle$ where L and M range over the left factors of E.)

Works because:

$$\mathsf{S}{\cdot}\mathsf{T} \subseteq [\![\mathsf{E}]\!] \quad \equiv \quad (\mathsf{S} \subseteq [\![\mathsf{E}]\!]) \bullet (\mathsf{T} \subseteq [\![\mathsf{E}]\!]) \quad .$$

where $\mathsf{B}{\bullet}\mathsf{C}$ denotes the composition of relations B and C.

Proof

We have to show that

$$S{\cdot}T \subseteq U\lhd\backslash W\lhd \quad\equiv\quad \langle\exists V :: S \subseteq U\lhd\backslash V\lhd \;\wedge\; T \subseteq V\lhd\backslash W\lhd\rangle \quad .$$

First,

$$\begin{array}{ll}
 & S{\cdot}T \subseteq E \\
= & \quad\{ \quad \text{unit of conjunction} \quad \} \\
 & S{\cdot}T \subseteq E \wedge \mathsf{true} \\
= & \quad\{ \quad \text{factors, } T\lhd = E/T\text{; cancellation} \quad \} \\
 & S \subseteq T\lhd \;\wedge\; T\lhd{\cdot}T \subseteq E \\
= & \quad\{ \quad \text{factors, } T\lhd\rhd = T\lhd\backslash E \quad \} \\
 & S \subseteq T\lhd \;\wedge\; T \subseteq T\lhd\rhd \quad .
\end{array}$$

Whence:

$$
\begin{array}{ll}
 & \mathsf{S}{\cdot}\mathsf{T} \subseteq \mathsf{U}{\lhd} \backslash W{\lhd} \\
= & \{ \quad \text{factors, definition of } W{\lhd} \quad \} \\
 & \mathsf{U}{\lhd} \cdot \mathsf{S} \cdot \mathsf{T} \cdot W \subseteq \mathsf{E} \\
= & \{ \quad \text{above, with } \mathsf{S},\mathsf{T} := \mathsf{U}{\lhd} \cdot S \ , \ \mathsf{T}{\cdot}W \quad \} \\
 & \mathsf{U}{\lhd} \cdot \mathsf{S} \subseteq (\mathsf{T}{\cdot}W){\lhd} \ \wedge \ \mathsf{T}{\cdot}W \subseteq (\mathsf{T}{\cdot}W){\lhd}{\rhd} \\
= & \{ \quad \text{factors} \quad \} \\
 & \mathsf{S} \subseteq \mathsf{U}{\lhd} \backslash (\mathsf{T}{\cdot}W){\lhd} \ \wedge \ \mathsf{T} \subseteq (\mathsf{T}{\cdot}W){\lhd}{\rhd} / W \\
= & \{ \quad \mathsf{U}{\rhd}/W = \mathsf{U} \backslash W{\lhd} \quad \} \\
 & \mathsf{S} \subseteq \mathsf{U}{\lhd} \backslash (\mathsf{T}{\cdot}W){\lhd} \ \wedge \ \mathsf{T} \subseteq (\mathsf{T}{\cdot}W){\lhd} \backslash W{\lhd} \ . \\
\Rightarrow & \{ \quad \text{one-point rule} \quad \} \\
 & \langle \exists V \ :: \ \mathsf{S} \subseteq \mathsf{U}{\lhd} \backslash V{\lhd} \ \wedge \ \mathsf{T} \subseteq V{\lhd} \backslash W{\lhd} \rangle \\
\Rightarrow & \{ \quad \text{Leibniz} \quad \} \\
 & \langle \exists V \ :: \ \mathsf{S}{\cdot}\mathsf{T} \subseteq \mathsf{U}{\lhd} \backslash V{\lhd} \cdot V{\lhd} \backslash W{\lhd} \rangle \\
\Rightarrow & \{ \quad \text{cancellation,} \quad \} \\
 & \mathsf{S}{\cdot}\mathsf{T} \subseteq \mathsf{U}{\lhd} \backslash W{\lhd} \ .
\end{array}
$$

Summary

- Use of **fusion** as programming method.
- **Problem generalisation** involves generalising the **algebra** in the solution domain.
- **Factor theory** as basis for language inclusion problems.

Challenges

- Efficient computation of factor matrices.
- Extension to non-regular languages.

References

J.H. Conway, "Regular Algebra and Finite Machines", Chapman and Hall, London, 1971.

Backhouse, R.C and Carré, B.A. "Regular algebra applied to path-finding problems", J. Institute of Mathematics and its Applications, vol. 15, pp. 161–186, 1975.

Backhouse, R.C. and Lutz, R.K., "Factor graphs, failure functions and bi-trees", Automata, Languages and Programming, LNCS 52, pp. 61–75, 1977.

Roland Backhouse, "Fusion on Languages", 10th European Symposium on Programming, LNCS 2028, pp. 107–121, 2001.

O. de Moor, S. Drape, D. Lacey and G. Sittampalam. Incremental program analysis via language factors. (`www.comlab.ox.ac.uk/oucl/work/oege.demoor/pubs.htm`)

For related publications on fixed points, Galois connections and mathematics of program construction, see `www.cs.nott.ac.uk/~rcb/papers`