Expressiveness and Complexity of a Graph Logic

Anuj Dawar (Cambridge)

joint work with
Philippa Gardner (Imperial College) and Giorgio Ghelli (Pisa)

A View of Process Algebras

- A *term algebra* T given by a functional syntax.
- A *structural congruence* $\equiv$ on terms.
- A *reduction* or *evaluation* relation $\longrightarrow$.

One can also consider a *logic* for specifying and reasoning about properties of terms.

The logic should be invariant under the structural congruence $\equiv$.

Graph Algebra

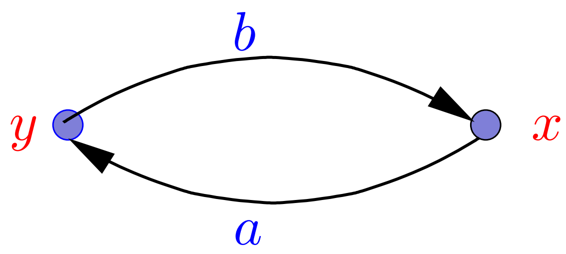
Corradini, Montanari and Rossi (1994) introduced a term language for describing graph structured data.

$$
\begin{aligned}
G \quad ::= \quad & \mathsf{nil} \\
& a(x, y) \\
& G \mid G \\
& (\mathsf{local}\ x)G
\end{aligned}
$$

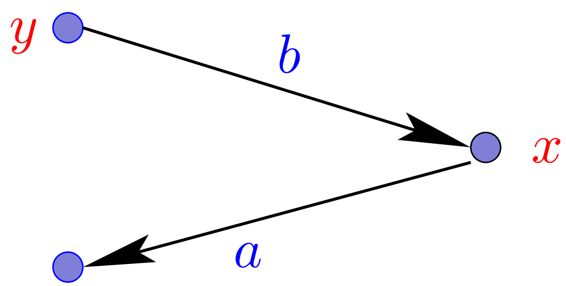
where $x, y \in \mathcal{X}$—a set of node names, and
$a \in \mathcal{A}$—a set of edge labels.

Examples

$a(x, y) \mid b(y, x)$



$(\mathsf{local}\ y)(a(x, y)) \mid b(y, x)$



Structural Congruence

The *structural congruence* is the least congruence closed with respect to $|$ and local and satisfying:

$$
\begin{aligned}
G|\mathsf{nil} &\equiv G \\
(G_1|G_2)|G_3 &\equiv G_1|(G_2|G_3) \\
G_1|G_2 &\equiv G_2|G_1 \\
(\mathsf{local}\ x)(\mathsf{local}\ y)G &\equiv (\mathsf{local}\ y)(\mathsf{local}\ x)G \\
(\mathsf{local}\ x)(G_1|G_2) &\equiv (\mathsf{local}\ x)G_1|G_2 \quad x \notin \mathsf{fn}(G_2) \\
(\mathsf{local}\ x)\mathsf{nil} &\equiv \mathsf{nil} \\
(\mathsf{local}\ x)G &\equiv (\mathsf{local}\ y)G\{y/x\}, \quad y \notin \mathsf{fn}(G)
\end{aligned}
$$

Variations

This structural congruence, given by Cardelli, Gardner and Ghelli (2001) corresponds to isomorphism under a *multiset* interpretation.

$a(x, y) \mid a(x, y)$ is a graph with two edges.

Other variations allow an interpretation without multiple edges, or one where structural congruence corresponds to bisimulation Buneman, Davidson, Hillebrand and Suciu (1996).

Graph Structures

Alternatively, a *graph structure* is given as:

$$(V \cup E \cup A, \mathsf{edge}, \mathsf{src} : X \to V)$$

where,

- V is a finite set of vertices, E a finite set of edges and A a finite set of labels. X is a set of names.
- $\mathsf{edge} : E \to A \times V \times V$ associates with each edge a label and a source and destination vertex.
- src associates a distinct vertex with each name in X.

Composition

We can define the operation of graph composition on such relational structures.

If

$$G_1 = (V_1 \cup E_1 \cup A_1, \mathsf{edge}_1, \mathsf{src}_1)$$

and

$$G_2 = (V_2 \cup E_2 \cup A_2, \mathsf{edge}_2, \mathsf{src}_2)$$

then, $G_1|G_2$ is obtained by taking the disjoint union of G_1 and G_2 *except* that for any name x we identify the vertices $\mathsf{src}_1(x)$ and $\mathsf{src}_2(x)$.

Graph Logic

The formulas of the *graph logic* of Cardelli, Gardner and Ghelli are built up from

- a set $\mathcal{X}$ of node names,
- a set $\mathcal{A}$ of label names,
- a set $V_{\mathcal{X}}$ of node variables,
- a set $V_{\mathcal{A}}$ of label variables and
- a set $V_{\mathcal{R}}$ of relational variables (each with an associated arity)

by the following rules:

Graph Logic (contd.)

nil	
true	
$\alpha(\xi_1, \xi_2)$	$\alpha \in \mathcal{A} \cup V_{\mathcal{A}}, \xi_i \in \mathcal{X} \cup V_{\mathcal{X}}$
$\xi_1 = \xi_2, \alpha_1 = \alpha_2$	$\alpha_i \in \mathcal{A} \cup V_{\mathcal{A}}, \xi_i \in \mathcal{X} \cup V_{\mathcal{X}}$
$\phi \mid \psi$	
$\phi \wedge \psi, \neg\phi$	
$\exists x.\phi, \exists a.\phi$	$x \in V_{\mathcal{X}}, a \in V_{\mathcal{A}}$
$R(\bar{\xi})$	$R \in V_{\mathcal{R}}$
$(\mu_{R,\bar{x}})\phi(\bar{\xi})$	R positive in ϕ

Semantics

$G \models_\sigma \mathsf{nil}$ iff $G \equiv \mathsf{nil}$

$G \models_\sigma \alpha(\xi_1, \xi_2)$ iff $G \equiv \sigma\alpha(\sigma\xi_1, \sigma\xi_2)$.

$G \models_\sigma (\phi \mid \psi)$ iff $G \equiv G_1 \mid G_2$ and $G_1 \models_\sigma \phi$ and $G_2 \models_\sigma \psi$.

μ is a standard least fixed point operator.

Expressiveness and Complexity

Combined (or model-checking) complexity:
What is the complexity of the satisfaction relation $G \models \phi$?

Data complexity:
Associate with each formula ϕ, the set $G_\phi = \{G \mid G \models \phi\}$. How complex can these sets be?

What is the relation between the expressive power of this logic and other standard logics: *second-order logic*, MSO, LFP?

Monadic Second-Order Logic

We define the *monadic second-order logic of graphs* by:

- $\mathsf{edge}(e, \alpha, \xi_1, \xi_2)$; $e_1 = e_2$; $\alpha_1 = \alpha_2$; $\xi_1 = \xi_2$;
- $\phi \wedge \psi$; $\neg\phi$.
- $\exists x.\phi$; $\exists a.\phi$; $\exists e.\phi$;
- $X(e)$; $\exists X.\phi$; where X is a *set variable* ranging over sets of edges.

MSO

If we consider the fragment of Cardelli *et al.*'s graph logic without the *least fixed point* operator, we have an easy translation into MSO.

The key step is

$$(\phi \mid \psi)^* = \exists X.[(\phi^*)^X \wedge (\psi^*)^{\neg X}]$$

Complexity of Second-Order Logic

We know:

- (by Fagin and Stockmeyer): A property of graphs is definable in *existential second-order logic* if, and only if, it is decidable in NP, and in *second-order logic* if, and only if, it is decidable in the polynomial hierarchy.
- *Monadic second-order logic* can express complete problems at every level of the polynomial hierarchy.
- There are problems of very low computational complexity that are not definable in MSO.
- The *combined complexity* of second-order logic is EXPTIME-complete, while that of MSO is PSPACE-complete.

Complexity of Graph Logic

The translation into MSO gives us upper bounds on the complexity of graph logic (without fixed points):

- For any formula ϕ of graph logic, the class of graphs G_ϕ is in the polynomial hierarchy.
- The combined complexity of graph logic is in PSPACE.

We prove corresponding *hardness* results:

- Graph logic can express complete problems at every level of the polynomial hierarchy.
- The combined complexity of graph logic is PSPACE-complete.

Separating from MSO

Conjecture: There are graph properties definable in MSO that are not definable in the graph logic (without fixed points).

Candidates: 3-colourability, Hamiltonicity.

Note: we can express that a graph is connected, 2-colourable or that there are two disjoint paths from $\mathbf{x}$ to $\mathbf{y}$.

We need techniques for proving these are not definable.

Games

We define an *Ehrenfeucht-style game* for the graph logic.

We associate with each formula ϕ, its rank (r, s, t) where r is the nesting depth of $|$, s is the nesting depth of label quantifiers and t is the nesting depth of node quantifiers in ϕ.

The two players, Spoiler and Duplicator, play a game of rank (r, s, t) on a board which consists of two graphs G_1, G_2 each with markers $a_1, \ldots, a_m$ on some of the labels and $p_1, \ldots, p_l$ on some of the nodes.

Games (contd.)

Three kinds of move:

- node move
- label move
- decomposition move

Spoiler wins at rank $(0, 0, 0)$ only if G_1 and G_2 each consist of a single edge, and are not isomorphic.

Evenness

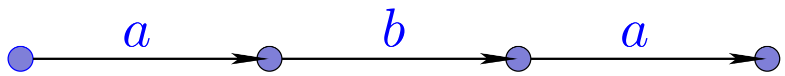
For each n, let S_n be the graph on $n+1$ nodes $\{c, v_1, \ldots, v_n\}$ with n edges $\{e_1, \ldots, e_n\}$ where e_i has the label a and connects c with v_i.

> For each k, and for all $n, n' > k2^k$ Duplicator has a winning strategy in the game played on S_n and $S_{n'}$ with rank (k, k, k).

There is no formula in the graph logic without recursion which expresses the property of having an even number of edges (or nodes).

Strings

Treating strings as a special kind of graph



We know that a language is expressible in MSO if, and only if, it is regular.

We show that every regular language is definable in the graph logic.

Regular Languages

We can write a formula that asserts that a graph *is a string* and one that asserts that a graph is a *disjoint collection of strings*.

G is a string in $L_1; L_2$ if there is a node x and a decomposition of G into two *strings* G_1 and G_2 with x the final node of G_1 and the initial node of G_2 *and* G_1 is a string in L_1 and G_2 is a string in L_2.

G is a string in L^* if there is a decomposition of G into two graphs, each of which is a *set of strings*, with each string being in L.

Recursion and Linear Composition

The *fixed point operator*, when combined with $|$ greatly increases the complexity of the logic.

A simpler logic was proposed which allows only *linear composition*.

$$\phi.|\psi$$

with $G \models \phi.|\psi$ *if, and only if:*

- $G \equiv G_1|G_2$
- $G_1 \models \phi$ and $G_2 \models \psi$ and
- G_1 consists of a single edge.

Recursion with Linear Composition

Linear composition appears to be a *first order* operation. A formula with only linear composition and no fixed-points can be translated into a first-order formula.

However, the logic with linear composition and the fixed-point operator is far more expressive than LFP.

In particular:

- we can express evenness;
- we can express NP-complete problems.

Work in Progress

- Showing that the graph logic without recursion is weaker than MSO.
- Showing that the two are equivalent over trees.
- Comparison with graph grammars.
- Establishing the exact complexity of graph logic with recursion.