Attribute Grammars and Categorical Semantics

Shin-ya Katsumata

Research Institute for Mathematical Sciences, Kyoto University
Kyoto, 606-8502, Japan sinya@kurims.kyoto-u.ac.jp

Abstract. We give a new formulation of attribute grammars (AG for short) called *monoidal AGs* in traced symmetric monoidal categories. Monoidal AGs subsume existing domain-theoretic, graph-theoretic and relational formulations of AGs. Using a 2-categorical aspect of monoidal AGs, we also show that every monoidal AG is equivalent to a synthesised one when the underlying category is closed, and that there is a sound and complete translation from local dependency graphs to relational AGs.

1 Introduction

Attribute grammars are a mechanism to assign computation with bidirectional information flow to derivation trees of context free grammars [18]. Our intention is to give a categorical formulation of AGs. We employ *traced symmetric monoidal categories* (TSMC for short) as the underlying categories for the formulation.

The key notion that links AGs and TSMCs is the circular (or recursive) computation.

Circular computation is tightly related to the characteristic feature of AGs, namely computation with bidirectional information flow. To illustrate this, we consider the situation that an AG assigns to a derivation tree (top left of Figure 1) a computation with bidirectional information flow (top right of Figure 1). Boxes P, Q, R are computation units assigned by the AG to nodes p, q, r in the tree. Depending on the configuration of P, Q, R, the entire computation may involve circular computation. For instance, on the bottom of Figure 1 the box P feed-backs the input from R to Q so that the entire computation has a cycle.

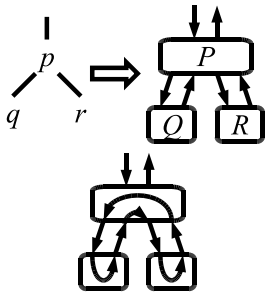

Fig. 1. Simple Description of Attribute Grammars

It is therefore natural to formulate AGs in a mathematical theory that admits circular computation. In [16], Joyal et al introduced the concept of *traced monoidal categories*. From the viewpoint of computer science, they provide an abstract account of feedback-loops, iteration and recursion in the models of computation, such as domain theory, iteration theory [5], Conway theory [6], relational models of flowcharts and networks [4], and so on.

The main observation of this paper is that by employing TSMCs as the underlying mathematical theory, we can achieve higher degree of abstraction in AGs. Following this observation, we propose a categorical formulation of AGs called *monoidal AGs*. The merit of this formulation is that we are free from concrete representation of data

structures and computation. We show that three existing formulations of AGs: 1) Chirica and Martin's K-systems [8], 2) Dependency graphs in classical AGs and 3) Courcelle and Deransart's relational AGs [9] are formally related to the instances of monoidal AGs. Subsequently, by exploiting a 2-categorical aspect of monoidal AGs we show that in closed TSMCs every monoidal AG is equivalent to the one which does not use inherited attributes, and that there exists a sound and complete translation from local dependency graphs into relational AGs. The latter result, which technically hinges on Selinger's work [21], appears to be new.

Preliminaries We adopt the standard algebraic treatment of CFGs. We regard a CFG $G = (T, N, S, P)$ as a many-sorted signature $\Sigma_G = (N, P)$ by identifying each production rule $p : X_0 \to v_0 X_1 v_1 \cdots X_n v_n \in P$ ($v_i \in T^*, X_i \in N, 0 \leq i \leq n$) and an operator $p : X_1 \cdots X_n \to X_0$. We also identify the set of derivation trees of G beginning with a non-terminal symbol $X \in N$ and the set $T_{\Sigma_G} X$ of closed Σ_G-terms of X. In this paper terminal symbols and the starting symbol do not play any role, so when declaring a CFG we just mention the set of nonterminal symbols and production rules.

The following concepts will be used in classical AGs. We fix a countably infinite set *Attr* of *attribute names*, and assume that it is closed under prefixing "$i.$" ($i \in \mathbf{N}$). A *named set* is a finite sequence of pairs of an attribute name and a set such that each attribute name in the sequence is different. For a named set $R = a_1 : V_1, \cdots, a_n : V_n$, by $|R|$ we mean $V_1 \times \cdots \times V_n$; for $x \in |R|$, by x_{a_i} we mean the i-th component of x; by $a(R)$ we mean $a_1, \cdots, a_n \in Attr^*$; by $n(R)$ we mean $\{a_1, \cdots, a_n\} \subseteq Attr$; by $i.R$ ($i \in \mathbf{N}$) we mean the named set $i.a_1 : V_1, \cdots, i.a_n : V_n$. For $l = a_1, \cdots, a_n \in Attr^*$ with distinct attribute names, by X^l we mean the named set $a_1 : X, \cdots, a_n : X$. For a pair of tuples a, b, by $a; b$ we mean the concatenation of them; for example, $(a, b); (c, d) = (a, b, c, d)$.

2 Classical AGs

We first informally describe the central idea of AGs. Let $G = (N, P)$ be a CFG. An AG $\mathcal{A}$ assigns a "computation unit" f_p (top of Figure 2) to each production rule $p : X_1 \cdots X_n \to X_0 \in P$. The computation unit has an I/O-port for X_0 at the top and n I/O-ports for $X_1 \cdots X_n$ at the bottom ($\mathcal{A}$ also specifies types of I/O ports, but we ignore them now). The unit processes all inputs and outputs simultaneously, regardless of direction. The information flowing downward is called *inherited attributes*, while the one flowing upward *synthesised attributes*. Given a derivation tree t of G, we construct a complex circuit by connecting computation units provided by $\mathcal{A}$ with each other according to the shape of t. The resulting circuit, which has an I/O port only at the top, is the computation assigned to t by the AG (bottom of Figure 2).

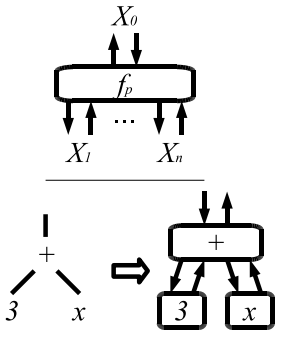

Fig. 2. Attribute Grammars

The above idea was proposed and formulated by Knuth in [18], where computation units were represented

by set-theoretic functions. Fix a CFG $G = (N, P)$. A *classical AG* for G is the tuple $\mathcal{A} = (\mathrm{I}, \mathrm{S}, f)$ where

1. I and S are N-indexed family of named sets such that $n(\mathrm{I}\,X) \cap n(\mathrm{S}\,X) = \emptyset$ for each $X \in N$. Below we write U_p for the named set $1.\,\mathrm{S}\,X_1, \cdots, n.\,\mathrm{S}\,X_n, \mathrm{I}\,X_0$ and D_p for the named set $\mathrm{S}\,X_0, 1.\,\mathrm{I}\,X_1, \cdots, n.\,\mathrm{I}\,X_n$.
2. f is a P-indexed family of functions such that $f_p : |\,\mathrm{U}_p\,| \to |\,\mathrm{D}_p\,|$ for each $p : X_1 \cdots X_n \to X_0 \in P$. They are called *attribute calculation rule*. By expanding the definition of U_p and D_p, we may also see f_p as the following function:

$$f_p : |\,\mathrm{S}\,X_1\,| \times \cdots \times |\,\mathrm{S}\,X_n\,| \times |\,\mathrm{I}\,X_0\,| \to |\,\mathrm{S}\,X_0\,| \times |\,\mathrm{I}\,X_1\,| \times \cdots \times |\,\mathrm{I}\,X_n\,|. \tag{1}$$

The assignment of computation to derivation trees is done by *meaning functions*. Let $\mathcal{A}$ be an AG for G. A N-indexed family of functions

$$\mathcal{A}[\![-]\!]_X : T_{\Sigma_G} X \to (|\,\mathrm{I}\,X\,| \to |\,\mathrm{S}\,X\,|) \tag{2}$$

(subscript X is often dropped) is called the *meaning function* of $\mathcal{A}$ if it satisfies the following condition: for any $p : X_1 \cdots X_n \to X_0 \in P$, $t_i \in T_{\Sigma_G} X_i$ $(1 \le i \le n)$ and $x \in |\,\mathrm{I}\,X_0\,|$, there exists $x_i \in |\,\mathrm{I}\,X_i\,|$ $(1 \le i \le n)$ such that

$$\mathcal{A}[\![\,p(t_1, \cdots, t_n)]\!](x); x_1; \cdots; x_n = f_p(\mathcal{A}[\![t_1]\!](x_1); \cdots; \mathcal{A}[\![t_n]\!](x_n); x). \tag{3}$$

Example 1. Consider a CFG G_p for expressions over integers:

$$G_p = (\{V, n, +\}, \{E\}, E, \{c_n : E \to n, \mathsf{var} : E \to V, \mathsf{plus} : E \to E + E\}),$$

where n ranges over $\mathbf{Z}$. The following data gives an AG $\mathcal{A}_p = (\mathrm{I}, \mathrm{S}, f)$ for G_p:

$$\mathrm{I}\,E = i : \mathbf{R}, \qquad \mathrm{S}\,E = s : \mathbf{R}$$

$$f_{c_n}(i) = n, \quad f_{\mathsf{var}}(i) = i, \quad f_{\mathsf{plus}}(1.s, 2.s, i) = (1.s + 2.s, i, i).$$

If a meaning function $\mathcal{A}_p[\![-]\!] : T_{\Sigma_{G_p}} E \to \mathbf{R} \to \mathbf{R}$ exists, then from (3) it satisfies

$$\mathcal{A}_p[\![c_n]\!](i) = n, \quad \mathcal{A}_p[\![\mathsf{var}]\!](i) = i, \quad \mathcal{A}_p[\![\mathsf{plus}(t, t')]\!](i) = \mathcal{A}_p[\![t]\!](i) + \mathcal{A}_p[\![t']\!](i).$$

Thus the meaning function of $\mathcal{A}_p$ evaluates expressions over integers with real numbers.

The problem of classical AGs is that the existence of meaning functions is not always guaranteed. This is technically because the witnesses $x_1, \cdots, x_n$ ensuring (3) may not exist under some situation. Another way to look at the problem is that the computation of value $\mathcal{A}[\![\,p(t_1, \cdots, t_n)]\!]$ requires feed-backs of the output $x_1, \cdots, x_n$ of f_p to itself, but such circular computation can not be modelled in a naive way using set-theoretic functions.

To resolve this problem, we shall either i) seek for AGs that do not induce circular computation (such AGs are called *non-circular* or *well-formed* [18, 10]) or ii) reformulate AGs within a mathematical theory that admits circular computation, such as domain theory. In this paper we take the latter option. For the mathematical foundation of the formulations of AGs, we employ *traced symmetric monoidal categories* [16, 14], which are recently recognised as providing an abstract representation of circular computation.

3 Traced Symmetric Monoidal Categories and Int Construction

We assume that readers are familiar with *symmetric monoidal categories* (*SMC* for short), *symmetric monoidal functors* and *monoidal natural transformations*; see e.g. [20]. We fix a common method for taking tensors of multiple objects in SMCs. Every SMC is equivalent to a strict one (coherence theorem [20]), so we mainly talk about strict SMCs for legibility. We reserve notations $\mathbf{I}, \otimes$ and c for the unit, tensor product and symmetry for SMCs, respectively.

In a SMC $\mathbb{C}$, one can represent a computation with n inputs and m outputs as a $\mathbb{C}$-morphism $f : A_1 \otimes \cdots \otimes A_n \to B_1 \otimes \cdots \otimes B_m$. In order to express feedback loops / circular computation under this representation, we adopt the concept of *trace operators*. They were originally introduced to balanced monoidal categories (which subsume SMCs) by Joyal et al in [16]. The following formulation of trace operators on SMCs is due to Hasegawa [14].

$$\begin{aligned}
&\mathbf{Tr}^{\mathbf{I}}_{A,B}(f) = f \\
&\mathbf{Tr}^{X\otimes Y}_{A,B}(f) = \mathbf{Tr}^{X}_{A,B}(\mathbf{Tr}^{Y}_{A\otimes X, B\otimes X}(f)) \\
&\mathbf{Tr}^{X}_{C\otimes A, C\otimes B}(C \otimes f) = C \otimes \mathbf{Tr}^{X}_{A,B}(f) \\
&\mathbf{Tr}^{X}_{X,X}(c_{X,X}) = \mathrm{id}_X \\
&\mathbf{Tr}^{X}_{A,B}(f \circ (g \otimes X)) = \mathbf{Tr}^{X}_{A',B}(f) \circ g \\
&\mathbf{Tr}^{X}_{A,B'}((g \otimes X) \circ f) = g \circ \mathbf{Tr}^{X}_{A,B}(f) \\
&\mathbf{Tr}^{X}_{A,B}((B \otimes g) \circ f) = \mathbf{Tr}^{Y}_{A,B}(f \circ (B \otimes g))
\end{aligned}$$

Fig. 3. Axioms for Trace Operators

Definition 1 ([16, 14]). *A* trace operator *on a SMC $\mathbb{C}$ is a family of mappings $\mathbf{Tr}^X_{A,B} : \mathbb{C}(A\otimes X, B\otimes X) \to \mathbb{C}(A, B)$ that satisfies the axioms summarised in Figure 3 (see [16, 14, 2] for graphical presentations of the axioms). A* traced symmetric monoidal category *(*TSMC*) is a pair of a SMC and a trace operator on it.*

Let $\mathbb{C}, \mathbb{D}$ be TSMCs. A traced symmetric monoidal functor *is a strong symmetric monoidal functor $(F : \mathbb{C} \to \mathbb{D}, m_I : \mathbf{I}_{\mathbb{D}} \xrightarrow{\cong} F\mathbf{I}_{\mathbb{C}}, m_{A,B} : FA \otimes_{\mathbb{D}} FB \xrightarrow{\cong} F(A \otimes_{\mathbb{C}} B))$ that preserves the trace operator in the following sense:*

$$(\mathbf{Tr}_{\mathbb{D}})^{FX}_{FA,FB}(m^{-1}_{A,B} \circ Ff \circ m_{A,B}) = F((\mathbf{Tr}_{\mathbb{C}})^{X}_{A,B}(f)).$$

Besides trace operators, in [16] Joyal et al gave a construction of categories called **Int**. It was originally considered for the structure theorem for traced balanced monoidal categories. In this paper **Int** construction will be used for obtaining the categories where computation with bidirectional information flow can be naturally modeled.

Definition 2 ([16]). *Let $\mathbb{C}$ be a TSMC. We define a category* **Int**$(\mathbb{C})$ *by the following data: an object is a pair (A^-, A^+) of $\mathbb{C}$-objects*[1]*, and a morphism $f : (A^-, A^+) \to (B^-, B^+)$ is a $\mathbb{C}$-morphism $f : A^+ \otimes B^- \to B^+ \otimes A^-$. The composition of f with $g : (B^-, B^+) \to (C^-, C^+)$ is defined to be the following morphism:*

$$\mathbf{Tr}^{B^-}_{A^+\otimes C^-, C^+\otimes A^-}((C^+ \otimes c) \circ (g \otimes A^-) \circ (B^+ \otimes c) \circ (f \otimes C^-) \circ (A^+ \otimes c)).$$

Consider a computation unit that has an input port A^+ and an output port A^- at the bottom, and an input port B^- and an output port B^+ at the top. This unit receives information from the bottom via A^+ and from the top via B^-, then outputs processed information to the bottom via A^- and to the top via B^+. In **Int**$(\mathbb{C})$ such a unit is expressed

[1] Compared to the original **Int** construction in [16], here the order of objects is swapped.

as a morphism $f : (A^-, A^+) \to (B^-, B^+)$, and its input-output relation is captured by a $\mathbb{C}$-morphism $f : A^+ \otimes B^- \to A^- \otimes B^+$. The definition of the composition in $\mathbf{Int}(\mathbb{C})$ is designed so that it correctly captures the input-output relation of the composition of two computation units ($\mathbf{Int}(\mathbb{C})$-morphisms); see [16, 1, 2] for graphical presentations of the composition.

Category $\mathbf{Int}(\mathbb{C})$ is a *compact closed category*, that is, a SMC such that every object has a left dual [17]. In this paper we only use the SMC structure of $\mathbf{Int}(\mathbb{C})$ given by

$$\mathbf{I}_{\mathbf{Int}(\mathbb{C})} = (\mathbf{I}, \mathbf{I}) \qquad (A^-, A^+) \otimes_{\mathbf{Int}(\mathbb{C})} (B^-, B^+) = (A^- \otimes B^-, A^+ \otimes B^+).$$

This tensor products correspond to combining I/O ports (and computation units) in parallel. For instance, the computation unit drawn on the top of Figure 2 can be expressed as an $\mathbf{Int}(\mathbb{C})$-morphism $f_p : (X_1^-, X_1^+) \otimes \cdots \otimes (X_n^-, X_n^+) \to (X_0^-, X_0^+)$.

Below we state the structure theorem for TSMCs. This is a specialisation of the one for traced balanced monoidal categories in [16].

Theorem 1. *The mapping $\mathbb{C} \mapsto \mathbf{Int}(\mathbb{C})$ can be extended to a left biadjoint to the forgetful functor from the 2-category of compact closed categories to that of TSMCs. The unit $\mathcal{N}_{\mathbb{C}} : \mathbb{C} \to \mathbf{Int}(\mathbb{C})$ of this biadjunction, which maps a $\mathbb{C}$-object A to an $\mathbf{Int}(\mathbb{C})$-object $(\mathbf{I}, A)$, is full and faithful.*

4 Monoidal AGs

In this section we give a categorical formulation of AGs, called *monoidal AGs*. We first introduce the concept of Σ-algebras for SMCs, which are a monoidal version of set-theoretic many-sorted algebras. We note that the concept of algebras in SMCs are also related to *operads* [19].

Definition 3. *Let $\Sigma = (S, O)$ be a signature and $\mathbb{C}$ be a SMC. A* Σ-algebra *in $\mathbb{C}$ is a pair (A, α) such that A is a S-indexed family of $\mathbb{C}$-objects and α is a O-indexed family of $\mathbb{C}$-morphisms such that $\alpha_o : As_1 \otimes \ldots \otimes As_n \to As$ for each $o : s_1 \ldots s_n \to s \in O$.*

Let $\mathcal{A} = (A, \alpha)$ be a Σ-algebra in $\mathbb{C}$. The meaning function *of $\mathcal{A}$ is a S-indexed family of mappings $\{\mathcal{A}[\![-]\!]_s : T_\Sigma s \to \mathbb{C}(\mathbf{I}, As)\}_{s \in S}$ such that the following holds for each $o : s_1 \cdots s_n \to s \in O$ (below we omit subscripts of meaning functions):*

$$\mathcal{A}[\![o(t_1, \cdots, t_n)]\!] = \alpha_o \circ (\mathcal{A}[\![t_1]\!] \otimes \cdots \otimes \mathcal{A}[\![t_n]\!]).$$

Definition 4. *A* monoidal AG *for a CFG $G = (N, P)$ in a TSMC $\mathbb{C}$ is a Σ_G-algebra $\mathcal{A} = (A, \alpha)$ in $\mathbf{Int}(\mathbb{C})$.*

This short and simple formulation captures essential information of AGs. We compare monoidal AGs and classical AGs below.

1. The set of sorts of Σ_G is N; so A assigns to each nonterminal symbol $X \in N$ an $\mathbf{Int}(\mathbb{C})$-object, say (A^-X, A^+X). We regard them as the domains of inherited and synthesised attributes respectively; so A plays the role of both I and S.

2. To each production rule $p : X_1 \cdots X_n \to X_0 \in P$, α assigns an **Int**($\mathbb{C}$)-morphism $\alpha_p : AX_1 \otimes \cdots \otimes AX_n \to AX_0$, which is the following $\mathbb{C}$-morphism by definition:

$$\alpha_p : A^+X_1 \otimes \cdots \otimes A^+X_n \otimes A^-X_0 \to A^+X_0 \otimes A^-X_1 \otimes \cdots \otimes A^-X_n.$$

   One can see the similarity between the domain and codomain of α_p and those of attribute calculation rule (1); here tensor products are used instead of direct products (this is the reason of the name "monoidal" AG).
3. The meaning function of a monoidal AG $\mathcal{A}$ for G is a mapping ($X \in N$)

$$\mathcal{A}[\![-]\!] : T_{\Sigma_G}X \to \mathbf{Int}(\mathbb{C})(\mathbf{I}, (A^-X, A^+X)) \cong \mathbb{C}(A^-X, A^+X),$$

   so it assigns to a derivation tree $t \in T_{\Sigma_G}X$ a computation from A^-X to A^+X expressed as a morphism in $\mathbb{C}$; compare this with (2).

To see the suitability of our categorical formulation of AGs, in the subsequent sections we compare instances of monoidal AGs and three existing formulations of AGs: i) Chirica and Martin's K-systems, ii) local dependency graphs in classical AGs and iii) Courcelle and Deransart's relational AGs.

4.1 Monoidal AGs in ωCPPO

The category ω**CPPO** of ω-complete pointed partial orders and continuous functions is Cartesian closed and has the least fixpoint operator $\mathbf{fix}_D : [[D \to D] \to D]$, which determines a trace operator:

$$\mathbf{Tr}^U_{AB}(f)(a) = \pi(\mathbf{fix}_{B\times U}(\lambda(b, u)\,.\ f(a, u)));$$

so ω**CPPO** is a traced CCC (for the above construction see [14]).

Monoidal AGs in ω**CPPO** are related to domain-theoretic formulations of AGs. Among various such formulations, here we establish a formal connection between Chirica and Martin's *K-systems* [8] and monoidal AGs. Fix a CFG $G = (N, P)$.

Definition 5 ([8]). *A* K-system *for G is a tuple $\mathcal{D} = (D^-, D^+, f)$ such that*

- *D^- and D^+ are N-indexed family of ω-CPPOs called* inherited *and* synthesised attribute domains, *respectively. For each $X \in N$, we write DX for $D^-X \times D^+X$.*
- *f is a P-indexed family of continuous functions such that for each $p : X_1 \cdots X_n \to X_0 \in P$, $f_p : [DX_0 \times DX_1 \times \cdots \times DX_n \to D^+X_0 \times D^-X_1 \times \cdots \times D^-X_n]$.*

A K-system assigns a continuous function $D^t : [D^-X \to D^+X]$ to a derivation tree $t \in T_{\Sigma_G}X$ ($X \in N$) as follows. We first recursively define a ω-CPPO D^t by

$$D^{p(t_1,\ldots,t_n)} = D^+X_0 \times D^-X_1 \times \cdots \times D^-X_n \times D^{t_1} \times \cdots \times D^{t_n} \qquad (p : X_1 \cdots X_n \to X_0 \in P)$$

For $d \in D^t$, by $\pi(d)$ we mean the first projection of d. Next, we construct a continuous function $H^t : [D^-X \times D^t \to D^t]$ by induction on the structure of t:

$$\begin{aligned} & H^{p(t_1,\ldots,t_n)}(i, (s, i_1, \ldots, i_n, w_1, \ldots, w_n)) \\ = \ & f_p((i, s), (i_1, \pi(w_1)), \ldots, (i_n, \pi(w_n))); (H^{t_1}(i_1, w_1), \ldots, H^{t_n}(i_n, w_n)). \end{aligned}$$

This function congregates one-step computation of inherited and synthesised attributes at every node of t. We then define the continuous function $\mathcal{D}^t : [D^- X \to D^+ X]$ that denotes the meaning of t by $\mathcal{D}^t(i) = \pi(\mathbf{fix}(\lambda x \in D^t \,.\, H^t(i, x)))$.

Let $\mathcal{D} = (D^-, D^+, f)$ be a K-system for G. We construct a monoidal AG $M(\mathcal{D}) = (D, \delta)$ for G in ω**CPPO** as follows:

$$DX = (D^- X, D^+ X)$$
$$\delta_p(s_1, \ldots, s_n, i) = \mathbf{fix}(\lambda(s, i_1, \ldots, i_n) \,.\, f_p((i, s), (i_1, s_1), \ldots, (i_n, s_n)))$$

where $X \in N$ and $p : X_1 \cdots X_n \to X_0 \in P$. On the other hand, every monoidal AG in ω**CPPO** can be casted to a K-system in an obvious way. These constructions preserve the meanings of Σ_G-terms.

Theorem 2. *Let $\mathcal{D}$ be a K-system for a CFG $G = (N, P)$ and $\mathcal{A}$ be a monoidal AG for G in ω**CPPO**. Then for any $t \in T_{\Sigma_G} X$ $(X \in N)$, we have $M(\mathcal{D})[\![t]\!] = \mathcal{D}^t$ and $(K(\mathcal{A}))^t = \mathcal{A}[\![t]\!]$.*

4.2 Monoidal AGs in Rel$^+$

The category **Rel** of sets and relations has Cartesian (bi)products, which, at object level, takes the disjoint sum of given sets. In [16] it was shown that the following is a trace operator with respect to the Cartesian products:

$$\mathbf{Tr}^U_{AB}(R) = R_{AB} \cup R_{UB} \circ (R_{UU})^* \circ R_{AU},$$

where R_{XY} $(X \in \{A, U\}, Y \in \{B, U\})$ is the restriction R to the relation between X and Y, and $(R_{UU})^*$ is the transitive reflexive closure of R_{UU}. The same operation was also considered in [4]. We call this TSMC **Rel**$^+$.

Monoidal AGs in **Rel**$^+$ are related to the concept of *local dependency graphs* (*LDG* for short) in classical AGs [18, 10]. Let $\mathcal{A} = (\mathrm{I}, \mathrm{S}, f)$ be a classical AG for a CFG $G = (N, P)$. We look at the syntactic definition of f, and assign to each production rule $p : X_1 \cdots X_n \to X_0 \in P$ the following digraph α_p: the set of vertices is $n(\mathrm{U}_p) \cup n(\mathrm{D}_p)$, and there is an edge in α_p from $a \in n(\mathrm{U}_p)$ to $a' \in n(\mathrm{D}_p)$ if and only if the a'-component of the result of f_p depends on the a-component of f_p's input. We usually draw α_p so that $n(\mathrm{I}\, X_0, \mathrm{S}\, X_0)$ are placed at the top and $n(k.\,\mathrm{I}\, X_k, k.\,\mathrm{S}\, X_k)$ $(1 \le k \le n)$ are placed at the bottom. The family α of digraphs constructed from $\mathcal{A}$ is called the *LDG* of $\mathcal{A}$. For instance, the LDG β of the classical AG $\mathcal{A}_p$ in Example 1 is at the top of Figure 4.

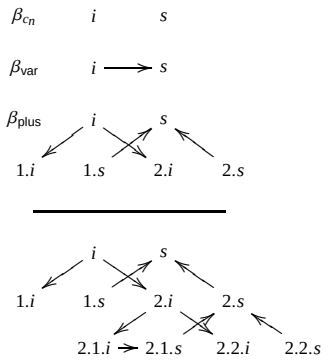

Fig. 4. LDG β of $\mathcal{A}_p$ (top) and an example of CDG (bottom)

LDGs are used to construct *compound dependency graphs* (*CDG* for short) of derivation trees. Let α be a LDG. For $t \in T_{\Sigma_G} X$

($X \in N$), we recursively construct a graph $\mathrm{CDG}_\alpha(t)$ as follows: $\mathrm{CDG}_\alpha(p(t_1, \cdots, t_n))$ is the union of α_p and the graphs obtained by adding a prefix "k." ($1 \le k \le n$) to every node in $\mathrm{CDG}_\alpha(t_k)$. In the bottom of Figure 4 $\mathrm{CDG}_\beta(\mathsf{plus}(c_3, \mathsf{plus}(\mathsf{var}, c_2)))$ is drawn. CDGs are a primary tool for detecting circular computation in classical AGs; see [18, 10].

By letting $AX = (n(\mathrm{I}\,X), n(\mathrm{S}\,X))$, each digraph α_p ($p : X_1 \cdots X_n \to X_0 \in P$) of a LDG α can be identified with a morphism in $\mathbf{Int}(\mathbf{Rel}^+)$:

$$\alpha_p \in \mathcal{P}(n(\mathrm{U}_p) \times n(\mathrm{D}_p)) \cong \mathbf{Int}(\mathbf{Rel}^+)(AX_1 \times \cdots \times AX_n, AX_0).$$

Thus a local dependency graph α of a classical AG for a CFG G specifies a monoidal AG $\overline{\alpha} = (A, \alpha)$ for G in $\mathbf{Rel}^+$.

Theorem 3. *Let α be a LDG of a classical AG $\mathcal{A}$ for a CFG $G = (N, P)$, and $t \in T_{\Sigma_G} X$ ($X \in N$). Then there exists a path from $i \in n(\mathrm{I}\,X)$ to $s \in n(\mathrm{S}\,X)$ in $\mathrm{CDG}_\alpha(t)$ if and only if $(i, s) \in \overline{\alpha}[\![t]\!]$.*

For example, $(i, s) \in \overline{\beta}[\![\mathsf{plus}(c_3, \mathsf{plus}(\mathsf{var}, c_2))]\!]$ as there is a path $i \to 2.i \to 2.1.i \to 2.1.s \to 2.s \to s$ in $\mathrm{CDG}_\beta(\mathsf{plus}(c_3, \mathsf{plus}(\mathsf{var}, c_2)))$, the graph on the bottom of Figure 4.

4.3 Monoidal AGs in $\mathbf{Rel}^\times$

The category **Rel** has another symmetric monoidal structure given by $A \otimes B = A \times B$ (Cartesian products of sets). This is a part of the compact closed structure over **Rel**, so **Rel** is canonically traced [16]; we call this TSMC $\mathbf{Rel}^\times$. The trace operator derived from the compact closed structure is the following:

$$\mathbf{Tr}^U_{AB}(R) = \{(a, b) \in A \times B \mid \exists u \in U\,.\,((a, u), (b, u)) \in R\}.$$

Monoidal AGs in $\mathbf{Rel}^\times$ are related to Courcelle and Deransart's *relational AGs* [9]. We first fix a many-sorted first-order logic $\mathcal{L}$ with a standard set-theoretic interpretation $[\![-]\!]$. For a typing context Γ of $\mathcal{L}$, by $i.\Gamma$ we mean the context obtained by adding a prefix "$i.$" to each variable in Γ. For a well-typed formula $\Gamma \vdash \Phi$, we define $i.\Phi$ to be the formula $\Phi[i.x/x]_{x \in \mathrm{var}(\Gamma)}$ (where $\mathrm{var}(\Gamma)$ is the set of variables in Γ). Clearly $i.\Gamma \vdash i.\Phi$.

Fix a CFG $G = (N, P)$.

Definition 6 ([9]). *A* relational AG *for G in $\mathcal{L}$ is a tuple $\mathcal{R} = (\Gamma, \Phi)$ such that*

- *Γ is a N-indexed family of typing contexts and*
- *Φ is a P-indexed family of formuli such that for each $p : X_1 \cdots X_n \to X_0 \in P$, Φ_p is the following well-typed formula:*

$$1.\Gamma X_1, \ldots, n.\Gamma X_n, \Gamma X_0 \vdash \Phi_p.$$

Let $\mathcal{R} = (\Gamma, \Phi)$ be a relational AG for G. For any $t \in T_{\Sigma_G} X$ ($X \in N$), we recursively define a formula $\Gamma X \vdash \Phi_t$ by

$$\Phi_{p(t_1, \ldots, t_n)} = \exists 1.\Gamma X_1, \ldots, n.\Gamma X_n.\Phi_p \wedge 1.\Phi_{t_1} \wedge \ldots \wedge n.\Phi_{t_n} \quad (p : X_1 \cdots X_n \to X_0 \in P).$$

We also define the relation $\mathcal{R}_t$ to be $[\![\Phi_t]\!]$.

From a relational AG $\mathcal{R} = (\Gamma, \Phi)$ for G in $\mathcal{L}$, we construct a monoidal AG in $\mathbf{Rel}^\times$ as follows. By letting $AX = (1, [\![\Gamma X]\!])$ for $X \in N$, we notice that the relation $[\![\Phi_p]\!]$ for each $p : X_1 \cdots X_n \to X_0 \in P$ can be identified with a morphism in $\mathbf{Int}(\mathbf{Rel}^\times)$:

$$[\![\Phi_p]\!] \in \mathcal{P}([\![1.\Gamma X_1, \ldots, n.\Gamma X_n, \Gamma X_0]\!]) \cong \mathbf{Int}(\mathbf{Rel}^\times)(AX_1 \otimes \ldots \otimes AX_n, AX_0).$$

Therefore $\mathcal{R}$ specifies a monoidal AG $\overline{\mathcal{R}} = (A, \alpha)$ where $\alpha_p = [\![\Phi_p]\!]$.

Theorem 4. *Let $\mathcal{R}$ be a relational AG for a CFG $G = (N, P)$ in $\mathcal{L}$. Then for any $X \in N$ and $t \in T_{\Sigma_G} X$, we have $\mathcal{R}_t = \overline{\mathcal{R}}\,[\![t]\!]$.*

5 Relating Monoidal AGs

We next see that functors and natural transformations between TSMCs give translations of monoidal AGs and relations between such translations. We begin with some categorical aspects of algebras in SMCs. Fix a signature $\Sigma = (S, O)$.

Definition 7. *Let $\mathcal{A} = (A, \alpha)$ and $\mathcal{B} = (B, \beta)$ be Σ-algebras in a SMC $\mathbb{C}$. A* Σ-algebra homomorphism *from $\mathcal{A}$ to $\mathcal{B}$ is a S-indexed family $\{h_s : As \to Bs\}_{s \in S}$ of $\mathbb{C}$-morphisms satisfying $\beta_o \circ (h_{s_1} \otimes \ldots \otimes h_{s_n}) = h_s \circ \alpha_o$ for each $o : s_1 \cdots s_n \to s \in O$. We write $\mathbf{Alg}_\Sigma(\mathbb{C})$ for the category of Σ-algebras and Σ-algebra homomorphisms in $\mathbb{C}$.*

We write **SMC** for the 2-category of small SMCs, symmetric monoidal functors and monoidal natural transformations. One can easily check that the mapping $\mathbb{C} \mapsto \mathbf{Alg}_\Sigma(\mathbb{C})$ extends to a 2-functor $\mathbf{Alg}_\Sigma : \mathbf{SMC} \to \mathbf{Cat}$. We note that set-theoretic Σ-algebras are precisely captured by $\mathbf{Alg}_\Sigma(\mathbf{Set})$, where **Set** is the category of sets and functions with tensors given by Cartesian products.

The meaning function in Definition 3 can be seen as *initial algebra semantics*. We write $\mathcal{T}_\Sigma = (T_\Sigma, \iota)$ for the initial object in $\mathbf{Alg}_\Sigma(\mathbf{Set})$.

Definition 8.
1. *Let $\mathbb{C}$ be a SMC. We define a symmetric monoidal functor $G_\mathbb{C} : \mathbb{C} \to \mathbf{Set}$ by $G_\mathbb{C} = \mathbb{C}(\mathbf{I}, -)$.*
2. *Let $\mathbb{C}, \mathbb{D}$ be SMCs and $(F : \mathbb{C} \to \mathbb{D}, m_I : \mathbf{I}_\mathbb{D} \to F\mathbf{I}_\mathbb{C}, m_{A,B} : FA \otimes_\mathbb{D} FB \to F(A \otimes_\mathbb{C} B))$ be a symmetric monoidal functor. We define a monoidal natural transformation $G_F : G_\mathbb{C} \to G_\mathbb{D} \circ F$ by $(G_F)_C(f) = Ff \circ m_I$.*
3. *For a Σ-algebra $\mathcal{A}$ in a SMC $\mathbb{C}$, by $|\mathcal{A}|$ we mean the underlying set-theoretic Σ-algebra $\mathbf{Alg}_\Sigma(G_\mathbb{C})(\mathcal{A})$. We say that two Σ-algebras in $\mathbb{C}$ are* equivalent *if their underlying algebras are isomorphic in $\mathbf{Alg}_\Sigma(\mathbf{Set})$.*

The meaning function $\mathcal{A}[\![-]\!]$ of a Σ-algebra $\mathcal{A}$ in a SMC $\mathbb{C}$ is equal to the unique morphism $! : \mathcal{T}_\Sigma \to |\mathcal{A}|$ in $\mathbf{Alg}_\Sigma(\mathbf{Set})$. We regard two equivalent algebras as giving the same meaning to Σ-terms, because their meaning functions are equal modulo an isomorphism.

The general algebraic concepts above will be used as follows. Let G be a CFG. The mapping $\mathbf{AG}_G : \mathbb{C} \mapsto \mathbf{Alg}_{\Sigma_G}(\mathbf{Int}(\mathbb{C}))$ is the construction of the *category of monoidal AGs* for G in a TSMC $\mathbb{C}$. It extends to a 2-functor $\mathbf{AG}_G$ from the 2-category of TSMCs to **Cat**, so relationships between TSMCs will immediately be reflected to those between

different formulations of AGs. Below we apply this fact to show that i) in closed TSMCs every monoidal AGs are equivalent to those which do not use inherited attributes, and ii) there is a sound and complete translation from LDGs to relational AGs.

5.1 Equivalence between Monoidal AGs and Synthesised Ones

AGs that do not use inherited attributes are called *synthesised AG* (*S-AG* for short). In [8] Chirica and Martin showed that by using function spaces as attribute domains, every K-system can be reduced to the one which does not use inherited attributes. This technique was also applied to the encoding of AGs by higher-order catamorphisms [11].

In this section we further generalise these results to monoidal AGs. We introduce a monoidal version of synthesised AGs (*monoidal S-AGs*), and show that in *closed* TSMCs every monoidal AG is equivalent to a monoidal S-AG. Fix a CFG $G = (N, P)$.

Definition 9. *A* monoidal S-AG *for G in a TSMC $\mathbb{C}$ is a monoidal AG (A, α) for G such that $A^- X = \mathbf{I}$ for each $X \in N$.*

It is easy to see that a monoidal AG $\mathcal{A}$ for G in a TSMC $\mathbb{C}$ is S-AG if and only if there exists a Σ_G-algebra $\mathcal{A}'$ in $\mathbb{C}$ such that $\mathcal{A} = \mathbf{Alg}_{\Sigma_G}(\mathcal{N}_{\mathbb{C}})(\mathcal{A}')$.

To show that *every* monoidal AG is equivalent to a monoidal S-AG, we need an extra structure on $\mathbb{C}$. Recall that a SMC $\mathbb{C}$ is *closed* if $- \otimes B$ has a right adjoint $B \multimap -$ for every $\mathbb{C}$-object B. The key of the equivalence is the following theorem due to Hasegawa.

Theorem 5 ([13]). *Let $\mathbb{C}$ be a TSMC. Then $\mathcal{N}_{\mathbb{C}} : \mathbb{C} \to \mathbf{Int}(\mathbb{C})$ has a symmetric monoidal right adjoint $\mathcal{R}_{\mathbb{C}} : \mathbf{Int}(\mathbb{C}) \to \mathbb{C}$ if and only if $\mathbb{C}$ is closed.*

When $\mathbb{C}$ is closed, $\mathcal{R}_{\mathbb{C}}$ can be defined by $\mathcal{R}_{\mathbb{C}}(A^-, A^+) = A^- \multimap A^+$.

Theorem 6. *Let $\mathcal{A}$ be a monoidal AG for G in a closed TSMC $\mathbb{C}$. Then $\mathcal{A}$ is equivalent to the monoidal S-AG $\mathbf{Alg}_{\Sigma_G}(\mathcal{N}_{\mathbb{C}})(\mathbf{Alg}_{\Sigma_G}(\mathcal{R}_{\mathbb{C}})(\mathcal{A}))$.*

5.2 A Translation from Local Dependency Graphs to Relational AGs

Two TSMCs $\mathbf{Rel}^+$ and $\mathbf{Rel}^\times$, which provide the underlying category for local dependency graphs and relational AGs, are linked by the finite multiset endofunctor $\mathcal{M} : \mathbf{Rel} \to \mathbf{Rel}$ defined as follows. First, we define the set MA to be the set of finite multisets of A. We identify an element of MA and a function $f \in A \to \mathbf{N}$ that returns non-zero at finitely many elements in A (so $MA = A \to \mathbf{N}$ if A is finite). We write $\{a\}$ ($a \in A$) for the function that returns 1 only at a. The endofunctor $\mathcal{M}$ is then defined by

$$\mathcal{M}A = MA, \qquad \mathcal{M}R = \{(h_1, h_2) \mid h \in MR\}$$

where $h_1(a) = \sum_{b \in B, (a,b) \in R} h(a, b)$ and $h_2(b) = \sum_{a \in A, (a,b) \in R} h(a, b)$. For any relation $R \subseteq A \times B$, we have $(a, b) \in R$ if and only if $(\{a\}, \{b\}) \in \mathcal{M}R$; so functor $\mathcal{M}$ is faithful.

Theorem 7 ([21]). *The above data gives a TSM functor $\mathcal{M} : \mathbf{Rel}^+ \to \mathbf{Rel}^\times$.*

Fix a CFG $G = (N, P)$. We consider the functor $\mathbf{AG}_G(\mathcal{M}) : \mathbf{AG}_G(\mathbf{Rel}^+) \to \mathbf{AG}_G(\mathbf{Rel}^\times)$ that gives a translation between monoidal AGs for G.

Proposition 1. *Let $\mathcal{A}$ be a monoidal AG for G in* $\mathbf{Rel}^+$. *Then $(i, s) \in \mathcal{A}[\![t]\!]$ if and only if $(\{i\}, \{s\}) \in (\mathbf{AG}_G(\mathcal{M})(\mathcal{A}))[\![t]\!]$.*

Functor $\mathbf{AG}_G(\mathcal{M})$ is the key of a sound and complete translation from LDGs to relational AGs. Let α be a LDG of a classical AG $\mathcal{A} = (\mathrm{I}, \mathrm{S}, f)$ for G. For each $p : X_1 \dots X_n \to X_0 \in P$, we define A_p by $A_p = 1.\mathrm{I}\,X_1, 1.\mathrm{S}\,X_1, \cdots, n.\mathrm{I}\,X_n, n.\mathrm{S}\,X_n, \mathrm{I}\,X_0, \mathrm{S}\,X_0$. Then we can encode $\mathcal{M}\alpha_p$ as a set of vectors of natural numbers:

$$\mathcal{M}\alpha_p \cong \{f \in |\mathbf{N}^{a(A_p)}| \mid \exists h \in \alpha_p \to \mathbf{N}. \bigwedge_{a \in n(\mathrm{D}_p)} f_a = h_1(a) \wedge \bigwedge_{a \in n(\mathrm{U}_p)} f_b = h_2(a)\}.$$

The relation on the right hand side can be expressed in a first-order logic with natural numbers and the standard interpretation of them (we only need the sort nat for natural numbers, logical connectives $\exists, \wedge, \top, =$ and constants $0, +$). Therefore from the LDG α we can construct a relational AG $\mathcal{R}_\alpha = (\Gamma, \Phi)$ as follows: for each $X \in N$, we define the context ΓX to be $i_1 : \mathrm{nat}, \cdots, i_n : \mathrm{nat}, s_1 : \mathrm{nat}, \cdots, s_m : \mathrm{nat}$ where $i_1, \cdots, i_n = a(\mathrm{I}\,X)$ and $s_1, \cdots, s_m = a(\mathrm{S}\,X)$, and we define the formula Φ_p ($p : X_1 \dots X_n \to X_0 \in P$) to be

$$\exists \{h_e\}_{e \in \alpha_p}. \left(\bigwedge_{a \in n(\mathrm{D}_p)} a = \sum_{b \in n(\mathrm{U}_p), (a,b) \in \alpha_p} h_{(a,b)} \right) \wedge \left(\bigwedge_{b \in n(\mathrm{U}_p)} b = \sum_{a \in n(\mathrm{D}_p), (a,b) \in \alpha_p} h_{(a,b)} \right).$$

For a named set R and $a \in n(R)$, by δ_a we mean the tuple $(0, \cdots, 0, \overset{a}{\check{1}}, 0, \cdots, 0) \in |\mathbf{N}^{a(R)}|$.

Theorem 8. *Let α be a LDG for a classical AG $\mathcal{A} = (\mathrm{I}, \mathrm{S}, f)$ of a CFG $G = (N, P)$. Then for any $t \in T_{\Sigma_G} X$ ($X \in N$), $i \in \mathrm{I}\,X$ and $s \in \mathrm{S}\,X$, $(\delta_i, \delta_s) \in (\mathcal{R}_\alpha)_t$ if and only if there exists a path from i to s in* $\mathrm{CDG}_\alpha(t)$.

6 Related Work

We have seen that our categorical formulation of AGs, namely monoidal AGs, are related to three existing formulations of AGs; K-systems [8], local dependency graphs [18] and Courcelle and Deransart's relational AGs [9]. However, there are many other formulations of AGs [7, 22, 15] that are not covered in this paper. The study of relationships between such AGs and monoidal AGs is left to the future work.

In [12] Girard proposed a novel interpretation of cut elimination called geometry of interaction (GoI), which was later analysed by researchers including Abramsky, Haghverdi, Jagadeesan and Scott [1–3]. It was revealed that TSMCs and **Int** construction were the key for an axiomatic account of GoI, and as a by-product many concrete TSMCs were investigated; see e.g. [1, 2]. It is interesting to examine if monoidal AGs in the TSMCs discovered in the study of GoI are useful in the applications of AGs, such as compiler constructions, program transformations and XML processing.

Acknowledgment I am indebted to Masahito Hasegawa for technical advises and stimulating discussions. I am also grateful to Susumu Nishimura, Keisuke Nakano, Kazuyuki Asada, Naohiko Hoshino and Ichiro Hasuo for valuable discussions.

References

1. Samson Abramsky. Retracting some paths in process algebra. In *Proc. CONCUR '96*, volume 1119 of *LNCS*, pages 1–17, 1996.
2. Samson Abramsky, Esfandiar Haghverdi, and Philip J. Scott. Geometry of interaction and linear combinatory algebras. *Math. Struct. in Comput. Sci.*, 12(5):625–665, 2002.
3. Samson Abramsky and Radha Jagadeesan. New foundations for the geometry of interaction. *Inf. Comput.*, 111(1):53–119, 1994.
4. E. S. Bainbridge. Feedbacks and generalized logic. *Inf. Control*, 31(1):75–96, 1976.
5. Stephen L. Bloom and Zoltán Ésik. *Iteration theories; the equational logic of iterative processes*. Springer-Verlag, 1993.
6. Stephen L. Bloom and Zoltán Ésik. Fixed-point operations on ccc's. part I. *Theor. Comput. Sci.*, 155(1):1–38, 1996.
7. John Boyland. Conditional attribute grammars. *ACM Trans. Program. Lang. Syst.*, 18(1):73–108, 1996.
8. Laurian M. Chirica and David F. Martin. An order-algebraic definition of Knuthian semantics. *Math. Sys. Theory*, 13:1–27, 1979.
9. Bruno Courcelle and Pierre Deransart. Proofs of partial correctness for attribute grammars with applications to recursive procedures and logic programming. *Inf. Comput.*, 78(1):1–55, 1988.
10. Pierre Deransart, Martin Jourdan, and Bernard Lorho. *Attribute Grammars; Definitions, Systems and Bibliography*, volume 323 of *LNCS*. Springer-Verlag, 1988.
11. Maarten Fokkinga, Johan Jeuring, Lambert Meertens, and Erik Meijer. A translation from attribute grammars to catamorphisms. *The Squiggolist*, 2(1):20–26, 1991.
12. Jean-Yves Girard. Geometry of Interaction I: Interpretation of System F. In R. Ferro et al., editor, *Logic Colloquium '88*. North-Holland, 1989.
13. Masahito Hasegawa. On traced monoidal closed categories. Invited talk at Traced Monoidal Categories, Network Algebras, and Applications 2007.
14. Masahito Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of* let *and* letrec. Springer-Verlag, 1999.
15. Bart Jacobs and Tarmo Uustalu. Semantics of grammars and attributes via initiality. In *Reflections on Type Theory, Lambda Calculus, and the Mind. Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday*, pages 181–196. Radboud University, 2007.
16. Andre Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.
17. G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.
18. Donald E. Knuth. Semantics of context-free languages. *Math. Sys. Theory*, 2(2):127–145, 1968. See *Math. Sys. Theory*, 5(1) 95-96, 1971 for a correction.
19. Tom Leinster. *Higher Operads, Higher Categories*, volume 298 of *London Math. Soc. Lecture Note Series*. Cambridge University Press, 2004.
20. Saunders MacLane. *Categories for the Working Mathematician (Second Edition)*, volume 5 of *Graduate Texts in Mathematics*. Springer, 1998.
21. Peter Selinger. A note on Bainbridge's power set construction. Manuscript, 1998.
22. S. Doaitse Swierstra and Harald Vogt. Higher order attribute grammars. In *Attribute Grammars, Applications and Systems*, volume 545 of *LNCS*, pages 256–296. Springer, 1991.