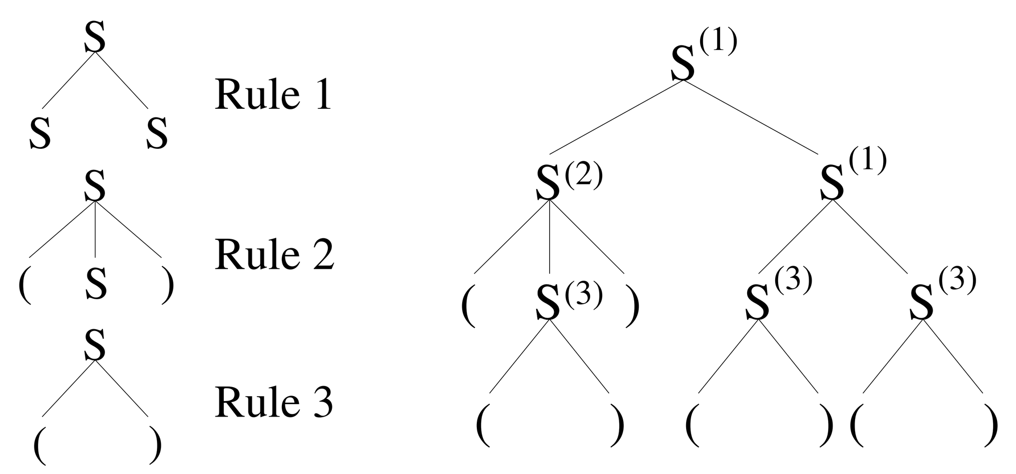
Representing Multidimensional Trees

David Brown, Ian Kelly, Colin Kern, Alex Lemann, Greg Sandstrom
Earlham College, Department of Computer Science

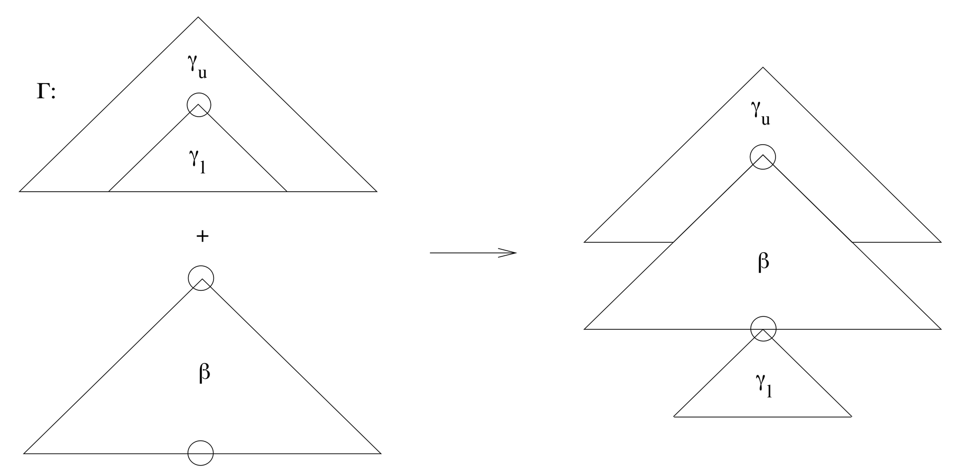
Abstract

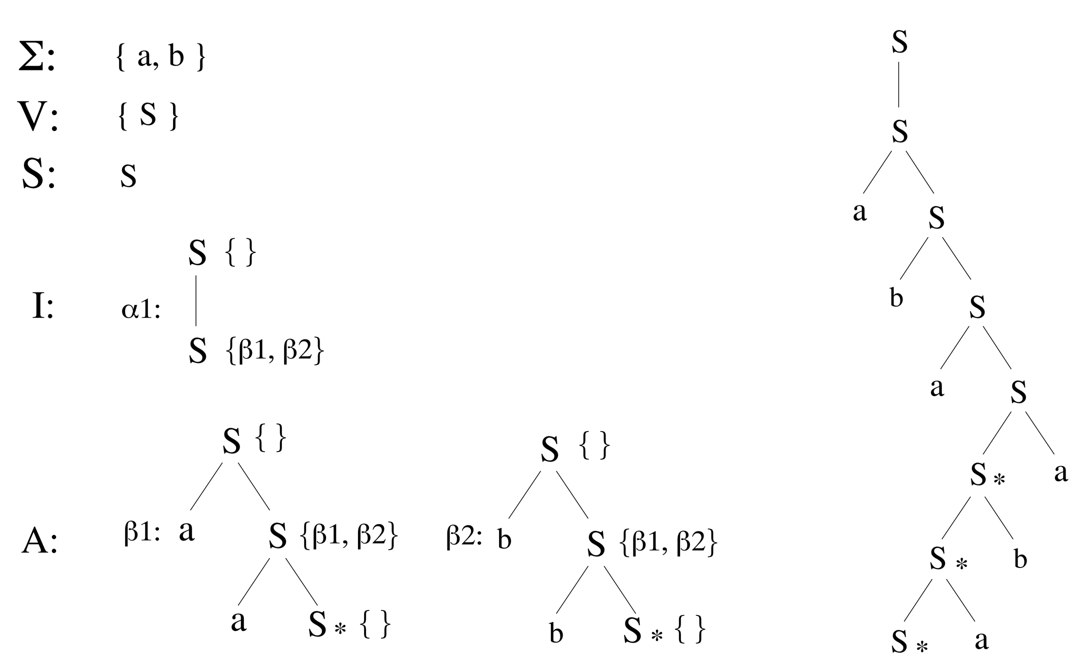
We develop a formal definition of multidimensional trees as abstract structures in "left-child, right-sibling" form. After developing this abstract definition, we show how it can be directly implemented as an ADT suitable for use in parsing applications. Additionally, we show how, when viewed in a slightly different way, our definition yields a flat form suitable for serialized input and output.

Context−Free Grammars (CFGs)

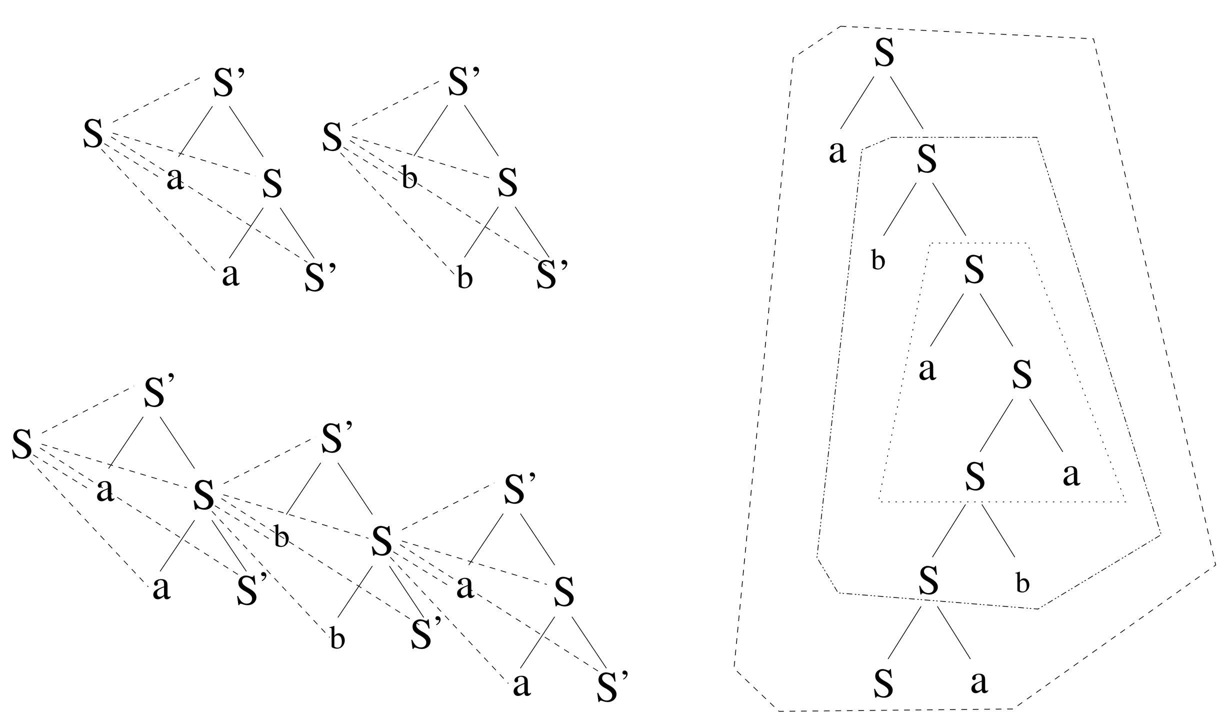


Tree Adjoining Grammars (TAG)

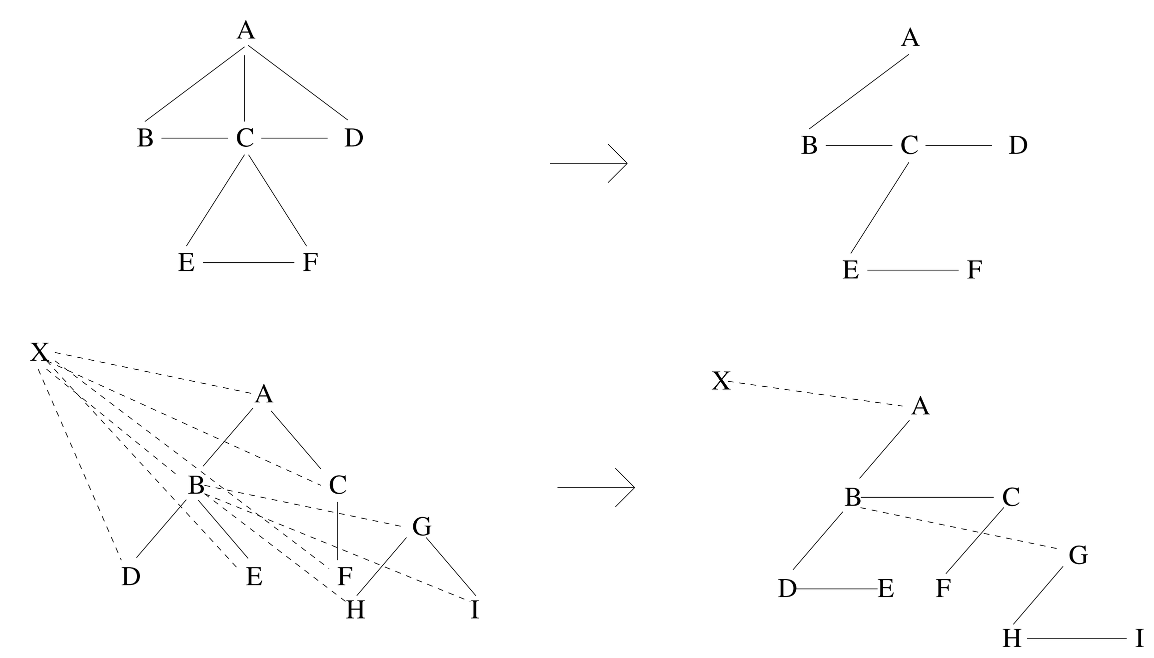




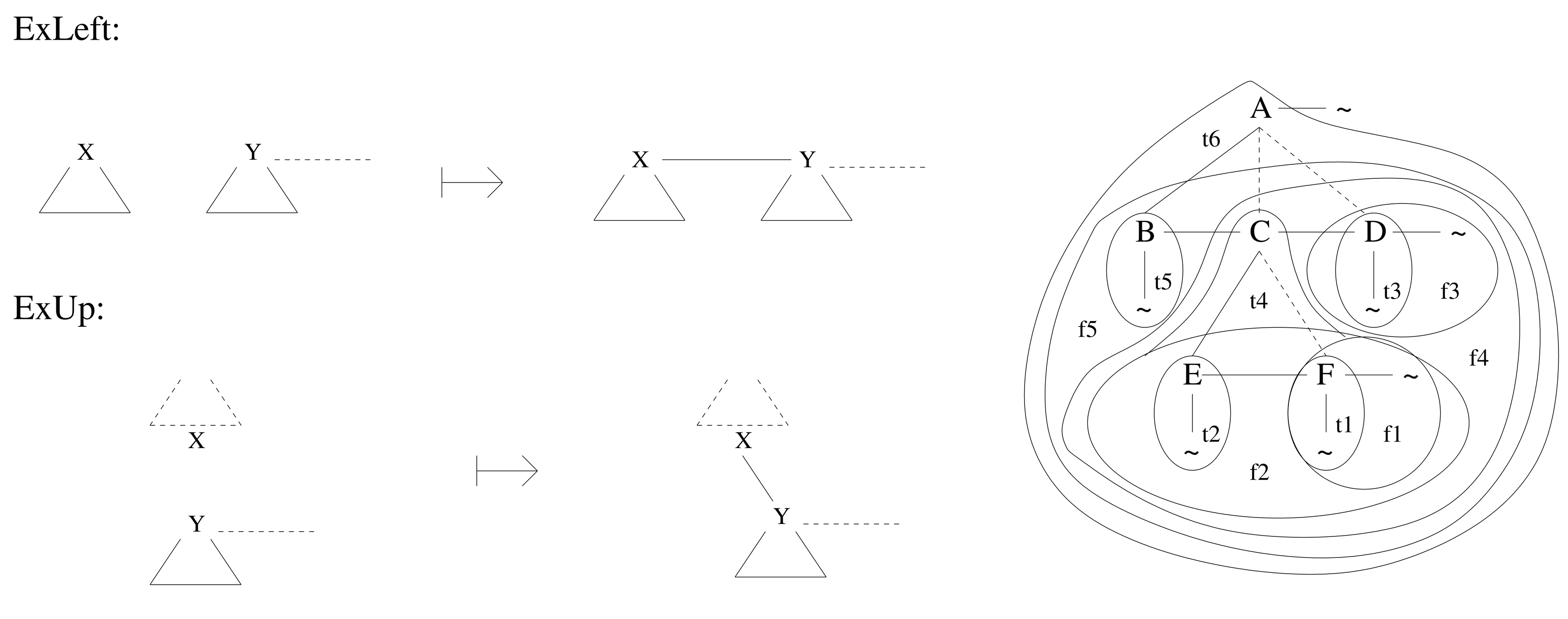
Multidimensional Grammars



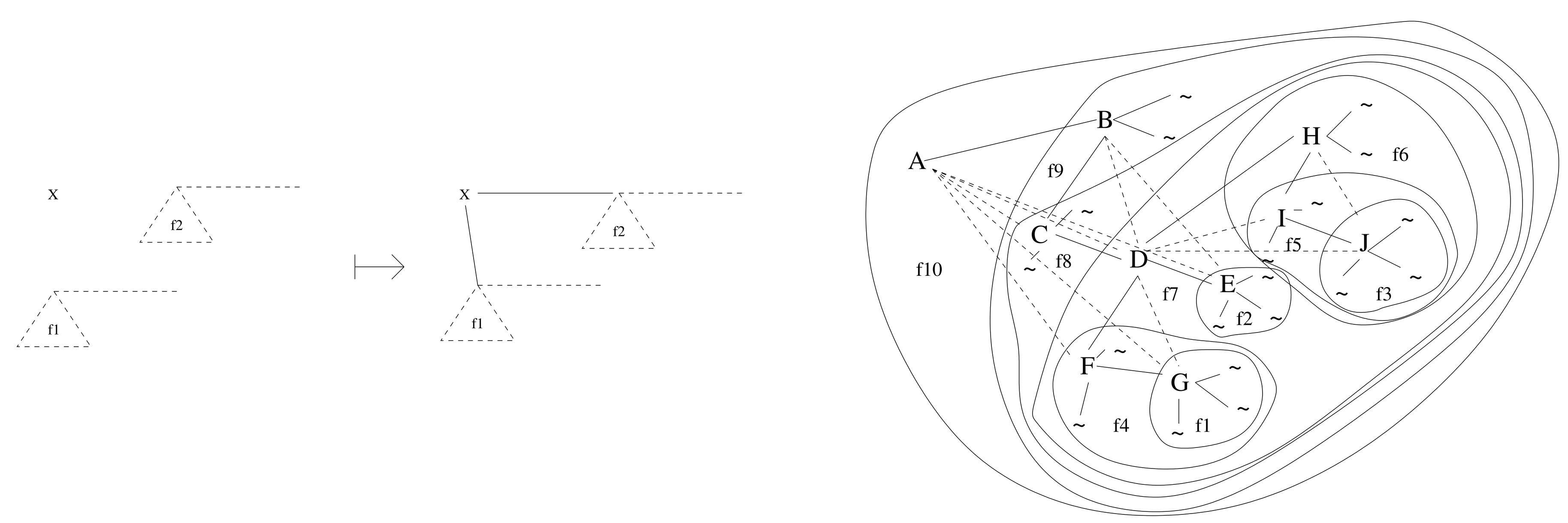
"Left−child, Right−sibling" Form



Two−Dimensional Constructor



Unified Multidimensional Constructor



(Preliminary) Tree-ordered Forests

- $\sim$ is an (empty) d-dimensional forest.
- If $t_1, t_2, \ldots, t_d$ are tree ordered forests and $X \in \Sigma$ then $T(X, t_1, t_2, \ldots, t_d)$ is a tree-ordered forest. t_i is the set of i-dimensional children of the new node labeled X.
- Nothing else is a tree-ordered forest.

Tree-ordered Forests—Fully Typed

- $\sim$ is an (empty) (i, d)-forest for all $0 \leq i \leq d$
- If $t_1, t_2, \ldots, t_d$ are, respectively, $(0, d)$-, $(1, d)$-, $\ldots$, $(d-1, d)$-forests and $X \in \Sigma$ then $T(X, t_1, t_2, \ldots, t_d)$ is a (j, d)-forest for all $0 \leq j \leq i$, where i is the smallest dimension such that t_i is not empty, or d if all t_k are empty. Here each t_k is the successor of the new node labeled X in the kth dimension.

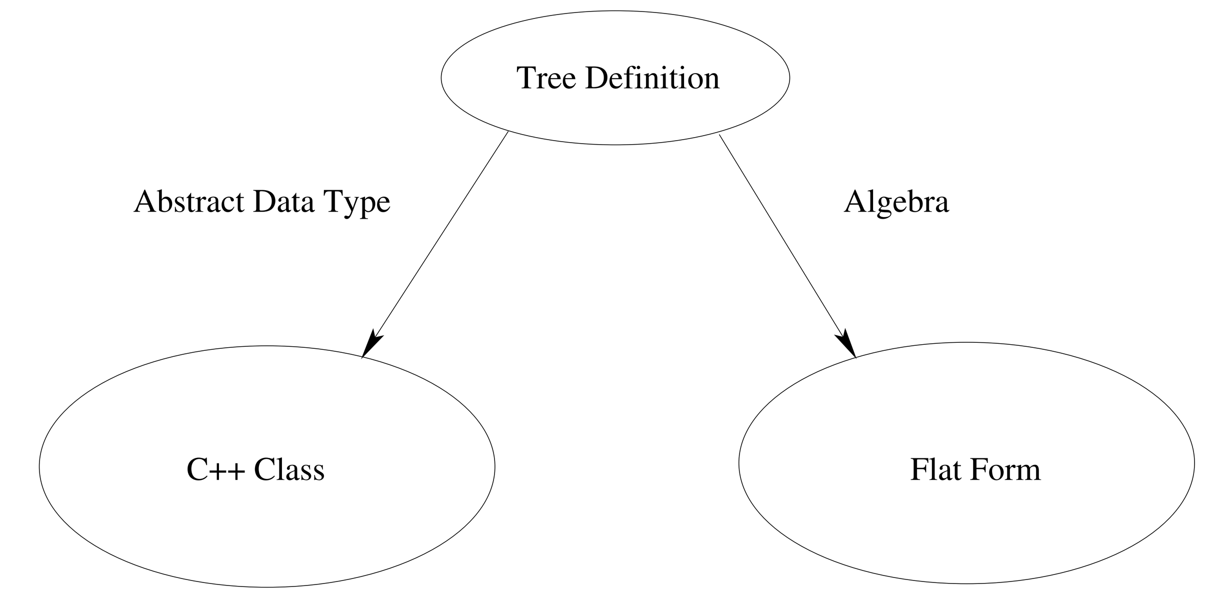
$(n-1)$-dimensional Local Yield

An $(n-1)$-dimensional local yield of a node is the set of all n-dimensional children of that node, which are ordered as an $(n-1)$-dimensional singleton forest, a tree—there must be exactly one minimum with respect to the $(n-1)$-dimensional ordering.

(i, d)-forests

Note that when assigning $f6$ as the 3-dimensional successor of D, the root of $f6$ does not have a 1-dimensional successor. In fact, it *can't* have a 1-dimensional successor—it is the root of a 2-dimensional local yield, which by our definition must be a singleton 2-dimensional forest. To denote a d-dimensional forest with a unique minimum (within a local structure, in this sense) in dimension i, we define an (i, d)-forest, where $0 \leq i \leq d$, as a forest whose root has an empty j-dimensional local yield for all $j < i$.

Concrete Forms



Flat Form

Definition

- $\sim$ is an (empty) (i, d)-forest in flat form for all $0 \leq i \leq d$
- If $t_1, t_2, \ldots, t_d$ are, respectively, $(0, d)$-, $(1, d)$-, $\ldots$, $(d-1, d)$-forests in flat form and $X \in \Sigma$ then $X(t_1, t_2, \ldots, t_d)$ is a (j, d)-forest in flat form for all $0 \leq j \leq i$, where i is the smallest dimension such that t_i is not empty, or d if all t_k are empty. Here each t_k is the successor of the new node labeled X in the kth dimension.
- Nothing else is a forest in flat form.

Flat Form Example

Example

The terms of the algebra for the two-dimensional tree to the left are:

$$F(\sim,\sim)$$
$$E(F(\sim,\sim),\sim)$$
$$D(\sim,\sim)$$
$$C(D(\sim,\sim),E(F(\sim,\sim),\sim))$$
$$B(C(D(\sim,\sim),E(F(\sim,\sim),\sim)),\sim)$$
$$A(\sim,B(C(D(\sim,\sim),E(F(\sim,\sim),\sim)),\sim))$$

Abstract Data Type Example (C++)

```
template<class label_type>
class forest
{
 public:
  forest(label_type label,                  /* A forest is a recursive structure:
         vector<forest*> links);                A node is a list of d links and a node label,
                                                where d is the dimensionality of the forest. */
  void set_label(label_type new_label);

  void set_link(std::size_t link_number,  /* Set link:
                forest* new_link);              Link accessors are parameterized by dimension. */

  label_type get_label( ) const;

  tree* get_successor(
         std::size_t link_number) const;

 private:
  label_type label;
  vector<forest*> link;
}
```

This research made possible in part by the Ford−Knight Research Grant, the Matthews Summer Research Fund, and the Earlham CS Collaborative Research Fund.