4.2: Isomorphism of Grammars

In this section, we study grammar isomorphism, i.e., the way in which grammars can have the same structure, even though they may have different variables.

Definition and Algorithm

Suppose G is the grammar with variables A and B, start variable A and productions:

$$\mathsf{A} \to \mathsf{0A1} \mid \mathsf{B},$$
$$\mathsf{B} \to \% \mid \mathsf{2A}.$$

And, suppose H is the grammar with variables B and A, start variable B and productions:

$$\mathsf{B} \to \mathsf{0B1} \mid \mathsf{A},$$
$$\mathsf{A} \to \% \mid \mathsf{2B}.$$

Question: how are G and H related?

Answer: H can be formed from G by renaming the variables A and B of G to B and A, respectively.

As a result, we say that G and H are isomorphic.

On the Subtlety of Isomorphism

Suppose G is as before, but that H is the grammar with variables 2 and A, start variable 2 and productions:

$$2 \to 021 \mid \mathrm{A},$$
$$\mathrm{A} \to \% \mid 22.$$

Then H can be formed from G by renaming the variables A and B to 2 and A, respectively.

Question: should we consider G and H to be isomorphic?

Answer: no—the symbol 2 is in both **alphabet** G and Q_H. In fact, G and H generate different languages.

A grammar's variables (e.g., A) can't be renamed to elements of the grammar's alphabet (e.g., 2).

Definition of Isomorphism

An *isomorphism* h from a grammar G to a grammar H is a bijection from Q_G to Q_H such that:

- h turns G into H;
- **alphabet** $G \cap Q_H = \emptyset$, i.e., none of the symbols in G's alphabet are variables of H.

We say that G and H are *isomorphic* iff there is an isomorphism between G and H.

As expected, we have that the relation of being isomorphic is reflexive on **Gram**, symmetric and transitive, and that isomorphism implies having the same alphabet and equivalence.

There is an algorithm for finding an isomorphism from one grammar to another, if one exists, or reporting that there is no such isomorphism. It's similar to the algorithm for finding an isomorphism between finite automata.

Renaming Variables

The function **renameVariables** takes in a pair (G, f), where G is a grammar and f is a bijection from Q_G to a set of symbols with the property that **range** $f \cap$ **alphabet** $G = \emptyset$, and returns the grammar produced from G by renaming G's variables using the bijection f. The resulting grammar will be isomorphic to G.

Renaming Variables

The following function is a special case of **renameVariables**. The function **renameVariablesCanonically** $\in$ **Gram** $\rightarrow$ **Gram** renames the variables of a grammar G to:

- A, B, etc., when the grammar has no more than 26 variables (the smallest variable of G will be renamed to A, the next smallest one to B, etc.); or
- $\langle 1 \rangle$, $\langle 2 \rangle$, etc., otherwise.

These variables will actually be surrounded by a uniform number of extra brackets, if this is needed to make the new grammar's variables and the original grammar's alphabet be disjoint.

Isomorphism Finding/Checking in Forlan

The Forlan module `Gram` contains the following functions for finding and processing isomorphisms in Forlan:

```
val isomorphism                  :
      gram * gram * sym_rel -> bool
val findIsomorphism              : gram * gram -> sym_rel
val isomorphic                   : gram * gram -> bool
val renameVariables              : gram * sym_rel -> gram
val renameVariablesCanonically : gram -> gram
```

Forlan Examples

Suppose the identifier `gram` of type `gram` is bound to the grammar with variables A and B, start variable A and productions:

$$A \to 0A1 \mid B,$$
$$B \to \% \mid 2A.$$

Suppose the identifier `gram'` of type `gram` is bound to the grammar with variables B and A, start variable B and productions:

$$B \to 0B1 \mid A,$$
$$A \to \% \mid 2B.$$

And, suppose the identifier `gram''` of type `gram` is bound to the grammar with variables 2 and A, start variable 2 and productions:

$$2 \to 021 \mid A,$$
$$A \to \% \mid 22.$$

Forlan Examples

Here are some examples of how the above functions can be used:

```
- val rel = Gram.findIsomorphism(gram, gram');
val rel = - : sym_rel
- SymRel.output("", rel);
(A, B), (B, A)
val it = () : unit
- Gram.isomorphism(gram, gram', rel);
val it = true : bool
- Gram.isomorphic(gram, gram'');
val it = false : bool
- Gram.isomorphic(gram', gram'');
val it = false : bool
```

Forlan Examples

```
- val gram = Gram.input "";
@ {variables} B, C
@ {start variable} B
@ {productions} B -> AC; C -> <A>
@ .
val gram = - : gram
- SymSet.output("", Gram.alphabet gram);
A, <A>
val it = () : unit
- Gram.output
= ("", Gram.renameVariablesCanonically gram);
{variables} <<A>>, <<B>> {start variable} <<A>>
{productions} <<A>> -> A<<B>>; <<B>> -> <A>
val it = () : unit
```