# Generative Code Modeling with Graphs

Marc Brockschmidt, Miltos Allamanis, Alexander L. Gaunt, Alex Polozov

Microsoft

Deep Program Understanding
https://aka.ms/MSRC-DPU

## Task

### Code Generation

Background:

- **Grammar** of target language known (used by tree generation approaches)
- Code **semantics** can be represented **as graph**
- **Attribute grammars** describe flow of information in code parsing as graph

## Key Ideas

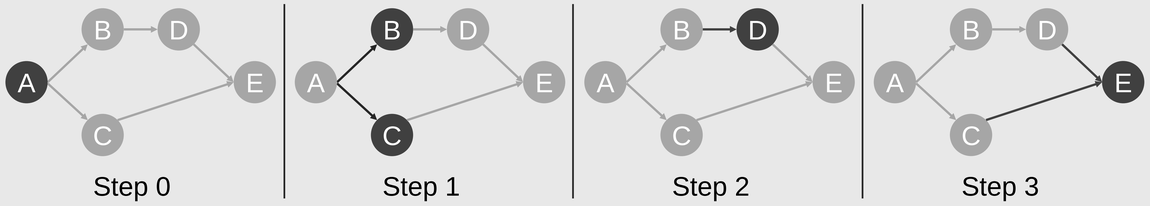
1. **Partially** generated **code** has **semantics**
2. **Neural Attribute Grammars** can learn semantics of partially processed code
3. **Asynchronous Graph Neural Networks** can propagate information in code generation order

## Asynchronous Graph Neural Networks (Liao et al, 2018)

**Observation:** Most GNNs are synchronous, update all node states at each time step

**Problem:** Computationally expensive for large & almost-sequential graphs

**Idea:** Define schedule of information propagation steps:



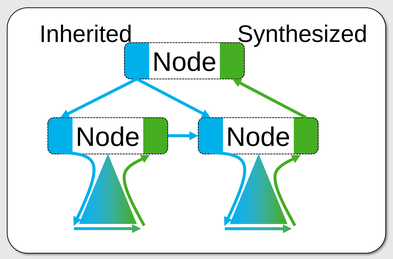

Graph Partition Neural Networks for Semi-Supervised Classification. R. Liao, M. Brockschmidt, D. Tarlow, A. Gaunt, R. Urtasun, R. Zemel (ICLR Workshop'18)

## Attribute Grammars

Concept from the (program) parsing literature

Core ideas:

- Nodes in abstract syntax trees (ASTs) have attributes
- **Inherited attributes:** Information from parents and preceding subtrees
- **Synthesized attributes:** Information about subtree




## Released Code

https://github.com/Microsoft/graph-based-code-modelling

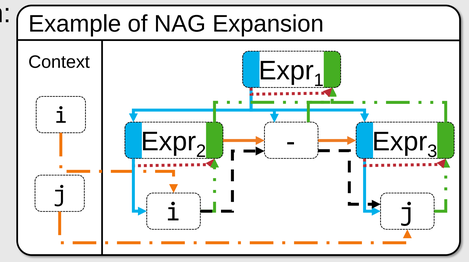
Included:

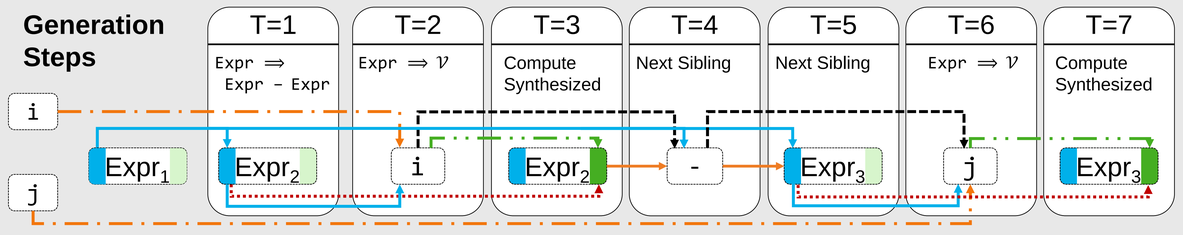
- Extracting program graphs & ASTs from C#
- Learning from programs with graphs (Allamanis et al, 2018)
- Code modeling with graphs in TensorFlow

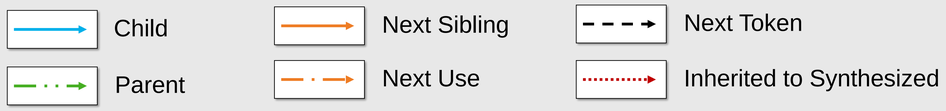
## Code Generation with Neural Attribute Grammars

Neural Attribute Grammars use AGs to structure generation:

- Generate top-down, left-right
- Each node expanded using grammar rule
- Edges represent flow of information (neural attributes)
- Child, parent edges + semantic edges
- Message propagation only for currently generated node
- Connect to context information




## Task & Evaluation

**Code Generation in Context**: Given a hole ■ in program fill it back in using just the context.

```
int methParamCount = 0;
if (paramCount > 0) {
    ...
}
if ( ■ ) {
        IParameterTypeInformation[] moduleParamArr =
            GetParamTypeInformations(Dummy.Signature,
                                     paramCount - methParamCount);
}
```

### Qualitative Examples

```
public static String URItoPath(String uri) {
   if (System.Text.RegularExpressions
           .Regex.IsMatch(uri, "^file:\\\\[a-z,A-Z]:")) {
       return uri.Substring(6);
   }
   if (■) {            // Ground Truth: uri.StartsWith(@"file")
      return uri.Substring(5);
   }
   return uri;
}
```

G → NAG
- ■ uri.Contains(UNK_STR)
- ■ uri.StartsWith(UNK_STR)
- ■ uri.HasValue()

G → Syn
- ■ uri == UNK_STR
- ■ uri == ""
- ■ uri.StartsWith(UNK_STR)

```
startPos = index + 1;
int count = endPos - startPos + 1;
word = (count > 0) ? ■ : String.Empty;
            // Ground Truth: input.Substring(startPos, count)
```

G → NAG
- ■ input + startPos
- ■ input + count
- ■ input.Substring(startPos, endPos – count)

G → ASN
- ■ input.Trim()
- ■ input.ToLower()
- ■ input + UNK_STR

### Within Projects

| Model | PPL | Type-Correct (%) | Exact Match @1 (%) |
|---|---|---|---|
| Seq→Seq | 87.4 | 32.4 | 21.8 |
| Seq→NAG | 6.8 | 53.2 | 17.7 |
| G→Seq | 93.3 | 40.9 | 27.1 |
| G→ASN | 2.6 | 78.7 | 45.7 |
| G→Syn | 2.7 | 84.9 | 50.5 |
| G→NAG | **2.6** | **86.4** | **52.3** |

### New Projects

| Model | PPL | Type-Correct (%) | Exact Match @1 (%) |
|---|---|---|---|
| Seq→Seq | 130.4 | 23.4 | 10.8 |
| Seq→NAG | 8.4 | 40.4 | 8.4 |
| G→Seq | 28.4 | 36.3 | 17.2 |
| G→ASN | 3.0 | 74.7 | 32.4 |
| G→Syn | **2.7** | **84.5** | 36.1 |
| G→NAG | 3.1 | **84.5** | **38.8** |