### 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:

1. **Step 0**
2. **Step 1**
3. **Step 2**
4. **Step 3**

### 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](https://github.com/Microsoft/graph-based-code-modelling)

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