

# A LANGUAGE AND COMPILER FOR GAME STRUCTURES

Brooks MacLachlan  
McMaster University

## Background

- A “game” is an activity involving some number of players and a way to “win”
  - Ex. sports, reality show games
- Infinitely many games are possible, but running most games would require a human “host”
- An easy-to-understand DSL for describing game structures and generating programs to run the game could reduce the need for human hosts and eliminate possibility of host biases affecting game results

## Related Work

- Existing game DSLs are focused on video games or computer players ([1], for example)
- Brantsteele is a website for simulating games [2]
- Few playable game variations available on websites such as Tengaged [3] and Zwooper [4]
- Online Reality Games (ORG) designed and run by human hosts are often played on social media platforms (see [5])

## Development Information

### The Program

- 1500+ lines of Haskell code
- Target language is Python

### Documentation

- All Haskell functions formally documented with Haddock

### Tests

- Parser, PreCompiler, and Compiler are fully unit-tested using the HSpec framework in Haskell
- Over 320 test cases in total!
- 6 additional full example games act as integration tests
  - Tennis, baseball, Survivor, Big Brother, The Genius, original

## Subset of the Grammar

$\langle game \rangle ::= \text{'Players:' } \langle teamList \rangle \text{' Rounds:' } \langle roundList \rangle \text{' Win:' } \langle winCondition \rangle$   
 $\langle competition \rangle ::= [\text{'scored'}] \text{ } [\text{'team'}] \text{' competition between' } \langle identifierList \rangle$   
 $\langle decision \rangle ::= \text{'vote by' } \langle identifierList \rangle \text{' between' } \langle identifierList \rangle [\text{'including self'}] \mid \dots$   
 $\langle affiliationUpdate \rangle ::= (\text{'add' } \mid \text{'remove'}) \langle name \rangle \mid \dots$   
 $\langle counterUpdate \rangle ::= \text{'set' } \langle name \rangle \text{' to' } \langle value \rangle \mid \dots$

## Example - DSL to Python

### Snippet of game description:

scored competition between everyone

**Parser:** Uses the Parsec library

```
competition = do {reserved "scored"
                  ; cmp <- competitor
                  ; reserved "competition"
                  ; reserved "between"
                  ; il <- identifierList
                  ; return $ Scored cmp il}
```

### AST node:

```
Scored Individual
  (IdList [IdVal Everyone (Num 1)] [])
```

### Compiler:

```
compileComp (Scored Individual il) = do
  ildoc <- compileIdentifierList il 1
  return $ (vcat [fst ildoc,
                 text "game.getScoredCompResults" <
                 parens (text "idList1")], snd ildoc)
```

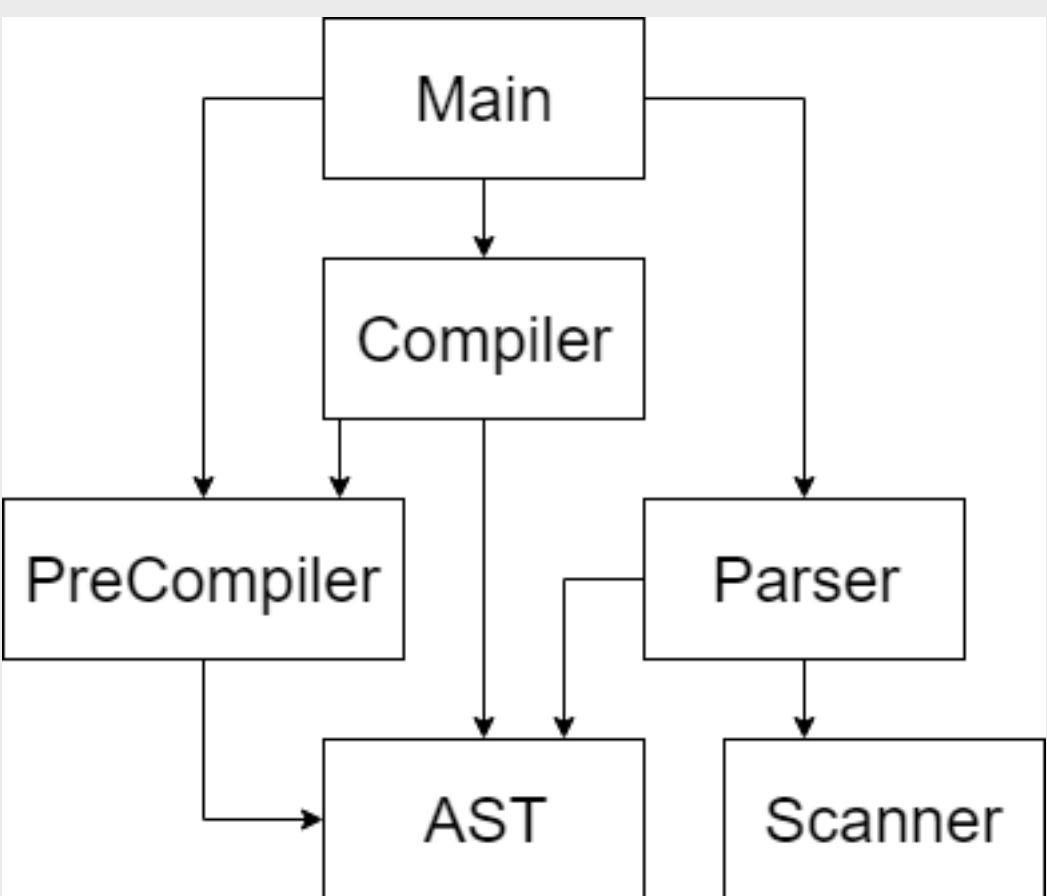
### Final Python code:

```
includeList1 = []; ident = game.playerList
idVal = ident; includeList1 += idVal
excludeList1 = []
idList1 = [x for x in includeList1
           if x not in excludeList1]
game.getScoredCompResults(idList1)
```

## References

- [1] Love, N., Hinrichs, T., Haley, D., Schkufza, E., Genesereth, M. (2008). General Game Playing: Game Description Language Specification. *Stanford Logic Group*.  
[2] Brantsteele. <https://brantsteele.com>  
[3] Tengaged. <https://tengaged.com>  
[4] Zwooper. <https://zwooper.com>  
[5] OnlineSurvivor. <https://www.reddit.com/r/OnlineSurvivor/>

## Modules



## Conclusion and Future Work

- The domain of game structures can be captured by a DSL, including well-known and completely original game structures
- The generated code in its current form is not particularly useful, improve with user interface, online-support, additional features such as game advantages, more conditional possibilities within rounds, variable names for phases, rounds, or tiebreakers