The Architects

SKYNET

Skynet - Basics

- Pure **Java** implementation
  - Based on Stanford's *Jocular* reference player
- Different strategies by varying the deployed gamer classes
  - SingleGamer: IDS with heuristics
  - MultiGamer: IDS, Minimax, prunning, heuristics
  - LegalGamer and
  - RandomGamer as fallbacks
- Gamer to be used is instantiated by a Factory with the help of the Game Information, deciding what type of player is needed
- Threaded searches for more precise clock-use (shorter buffer possible)

Skynet – Rule Analysis

- GameInformation class is also responsible for further analysis
- Number of players
- Successor relations (binary, bijective mapping)
- Step counter (unary, dependent on succ)
  - Used in single player games to see if certain goals can be reached at all
- Change of Control (limited)
  - No variables
  - Role names are used

Skynet – Rule Reordering

- Iteratively calculate the *domains* with the help of the dependency graph
- Sort the sentences by their *cardinality*
- Special case: (distinct [?]a [?]b)
  - Has to be put in a place where all variables used in distinct are already bound by preceding facts
- Special case: (does [?]rolename (...))
  - One player only has one move per turn
- Can be disabled (because it is still experimental)

Skynet - Heuristics

- Different heuristics can be chained together
- A special heuristics factory analyses the game information and instantiates that chain
- Implemented heuristics:
  - **Novelty**: weighted differences between old knowledge bases and future ones
  - **Mobility**: weighted number of moves
  - **SimpleGoalDistance**: number of grounded goal requirements true, weighted by their goal value