A Formalism for Build Order Search in StarCraft Brood War

Severin Wyss

Institute of Computer Science and Mathematics
University Basel

Bachelor Thesis, 2017

Why should you read this Thesis?

- Gives the formal basis for implementing Build Order planner for StarCraft Brood War.
- Formalism can also be used in other RTS. For example StarCraft 2.
- The description will allow to judge whether the formalism works with a other RTS.
- The concepts used to simplify the search space could also be useful in real life temporal planning.

Why StarCraft Brood War?

- There exists a community API for using AI agents directly in the original game: BWAPI.
- API allows to test AI agents versus human.
- Annually held tournaments between universities.
- One of the first and biggest competitive games. Therefore human skill and knowledge of domain is very strong.
- Now even more interesting: Blizzard and Deep Mind teams will enable AI agents in StarCraft 2 within 2017.

Real-Time-Strategy Game

A Real-Time-Strategy (RTS) game usually has the following structure:

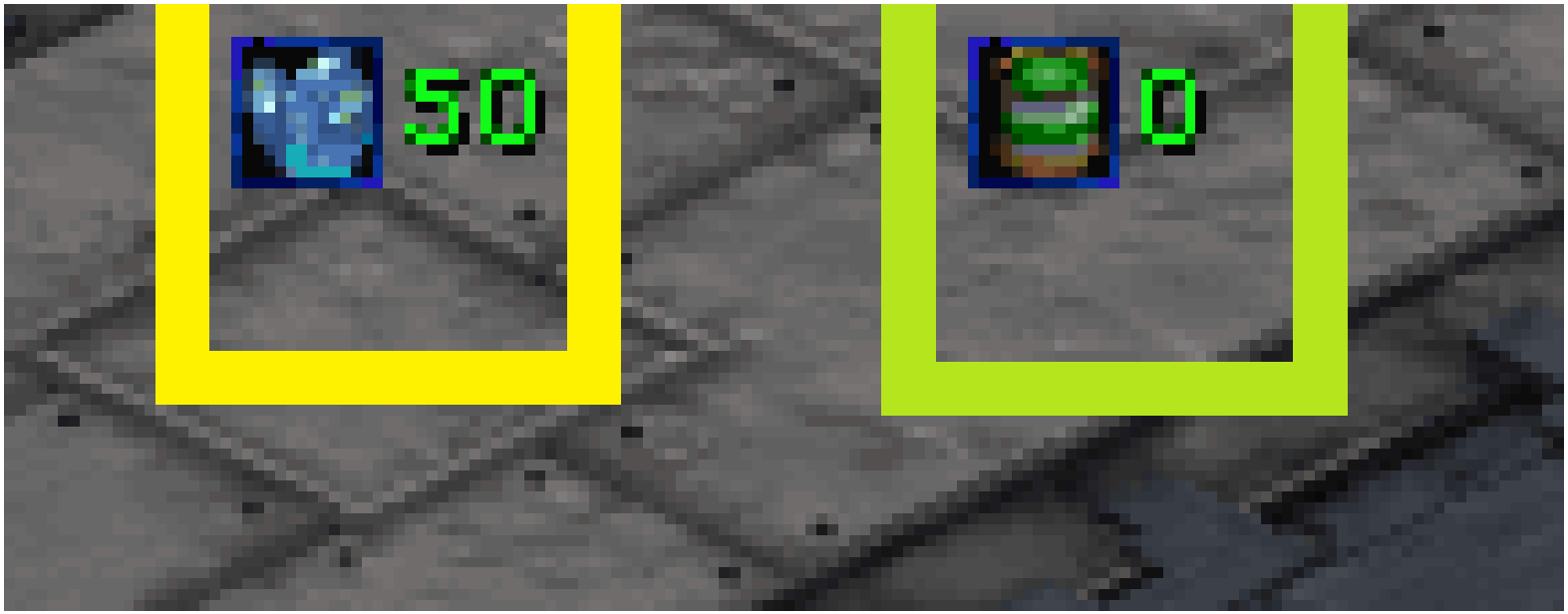
- Start with a few units and resources.
- Collect resources and build new units.
- When having build a reasonable army send units to attack the enemy.
- Fight the enemy.
- Win or lose the game.

We focus on the second item which is essentially about Build Orders.

StarCraft Brood War

Minerals, Gas

Minerals, Gas



Minerals and Gas are natural numbers greater than zero!
Example values: 0, 50, 400, 2500

States in SAS^+

From Foundations of AI course, we know what a state is in SAS^+.
States are a variable assignments such that each variable has a assignment.
Variable assignments to the variable v must be part of its finite domain $dom(v) = \{d_1, ..., d_n\}$.

Numerical Values

Variable assignments to the variable v in SAS$^+$ must be part of its finite domain $dom(v) = \{d_1, ..., d_n\}$.
Instead of a finite domain, we now can have infinite domains: $dom(v) = (R) \cup \infty$.
Additionally, effects and conditions include comparators ($<, =, \geq$ etc.) and computations ($+, -, \cdot$ etc.).

Main Building and Worker

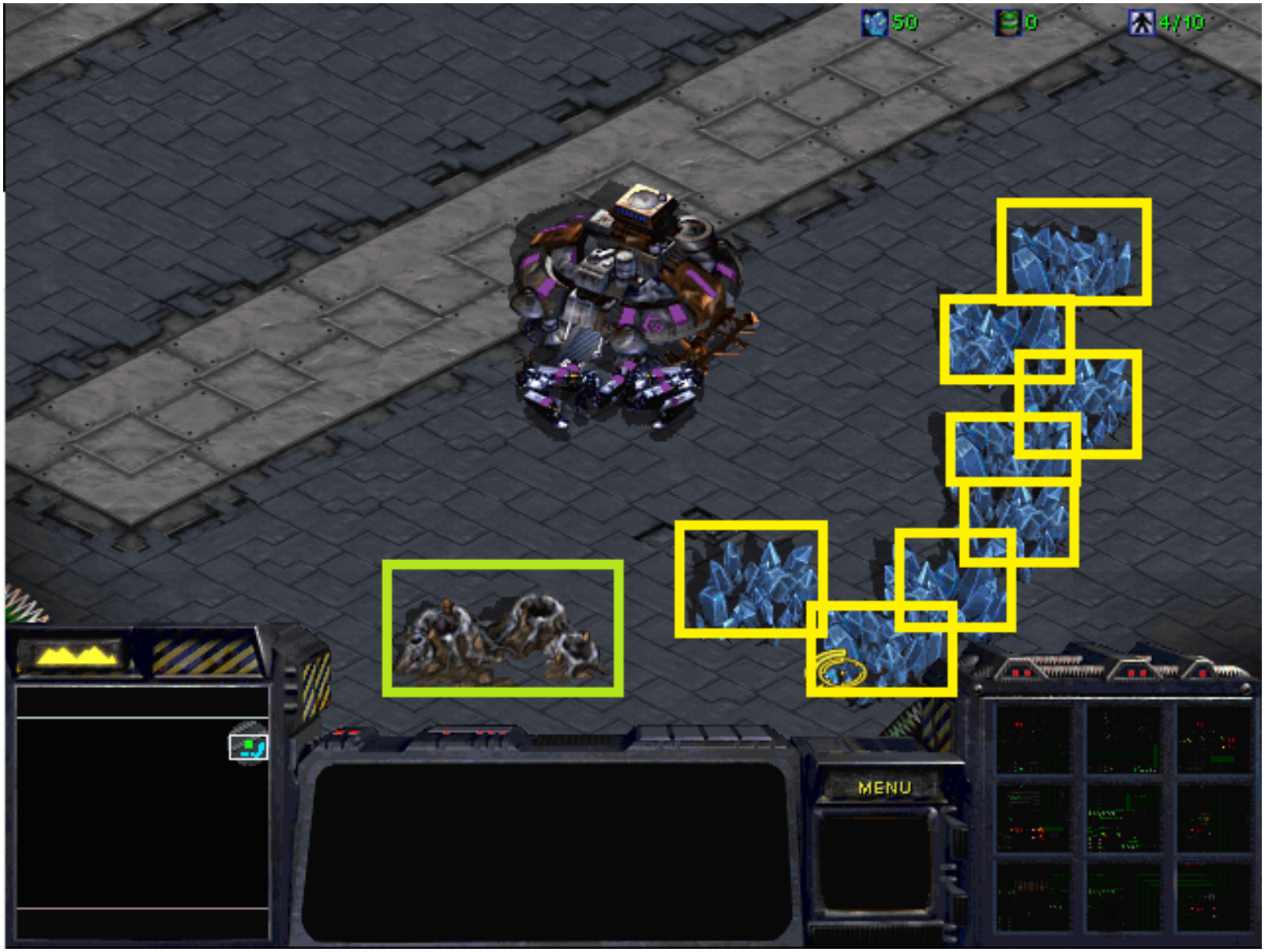
Units

SCV

Command Center

We will use units for the union of game units and game structures. Each unit has a set of task it can perform. Such as move, attack, gather resources, build new units etc.

Mineral Field and Vespine Gas Geyser



Tech Restriction

Tech Restriction

Yellow: can be build.

Gray: another unit is must exist first.

Actions in SAS^+

From Foundations of AI course, we know what a action is in SAS^+. Actions are a 3-tuple $a = \langle pre(a), eff(a), cost(a) \rangle$ where pre(a) and eff(a) are sets of variable assignments and cost(a) is a number.

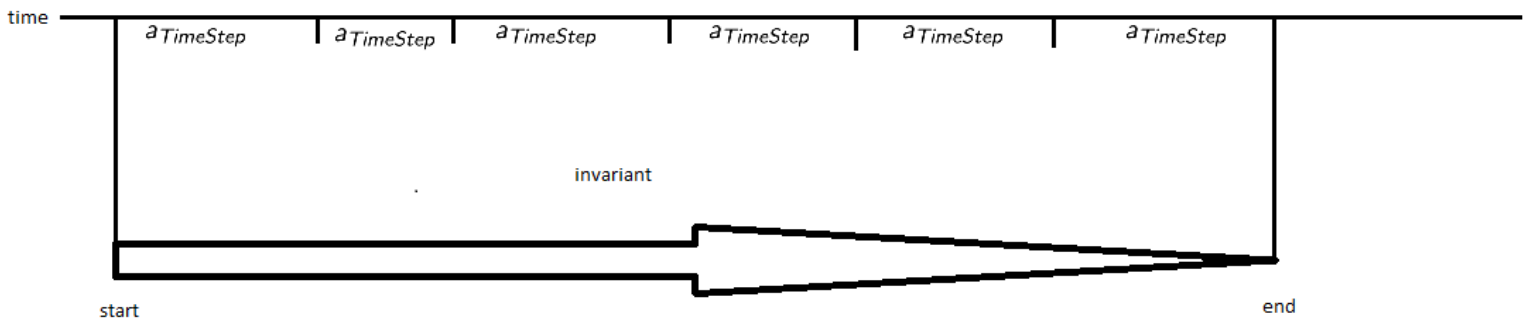
Temporal Action

Actions in SAS$^+$ are a 3-tuple $a = \langle pre(a), eff(a), cost(a) \rangle$
Temporal actions are 8-tuples

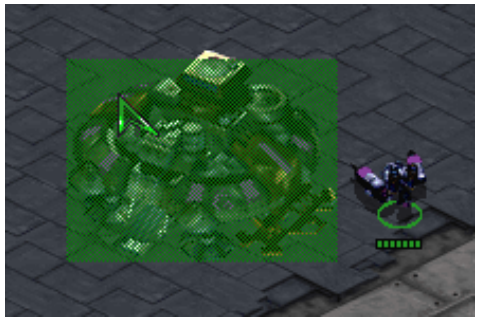
$$a_T = \langle d, pre_{start}(a_T), pre_{invar}(a_T), pre_{end}(a_T),$$
$$eff_{start}(a_T), eff_{invar}(a_T), eff_{end}(a_T), cost(a_T) \rangle$$

.

A special action is needed: $a_{TimeStep}$ which only advances time.



Building a Command Center



start (frame 0)

after $\sim$ 300 frames

after $\sim$ 800 frames

after $\sim$ 1700 frames

end (frame 1800)

State in our Formalism

A *State* is a 5-tuple $s := \langle f, U, R, m, g \rangle$

- f represents time
- m represents minerals
- g represents gas
- R are boolean values representing upgrades
- U is a set of units, each with their task

For example the initial state encodes as:

$$s_0 = (0, \{(Terran\_SCV, \emptyset, (IDLE, \infty), \infty, 4), \\ (Terran\_Command\_Center, \emptyset, (IDLE, \infty), \infty, 1)\} \\ , \{\}, 50.0, 0.0)$$

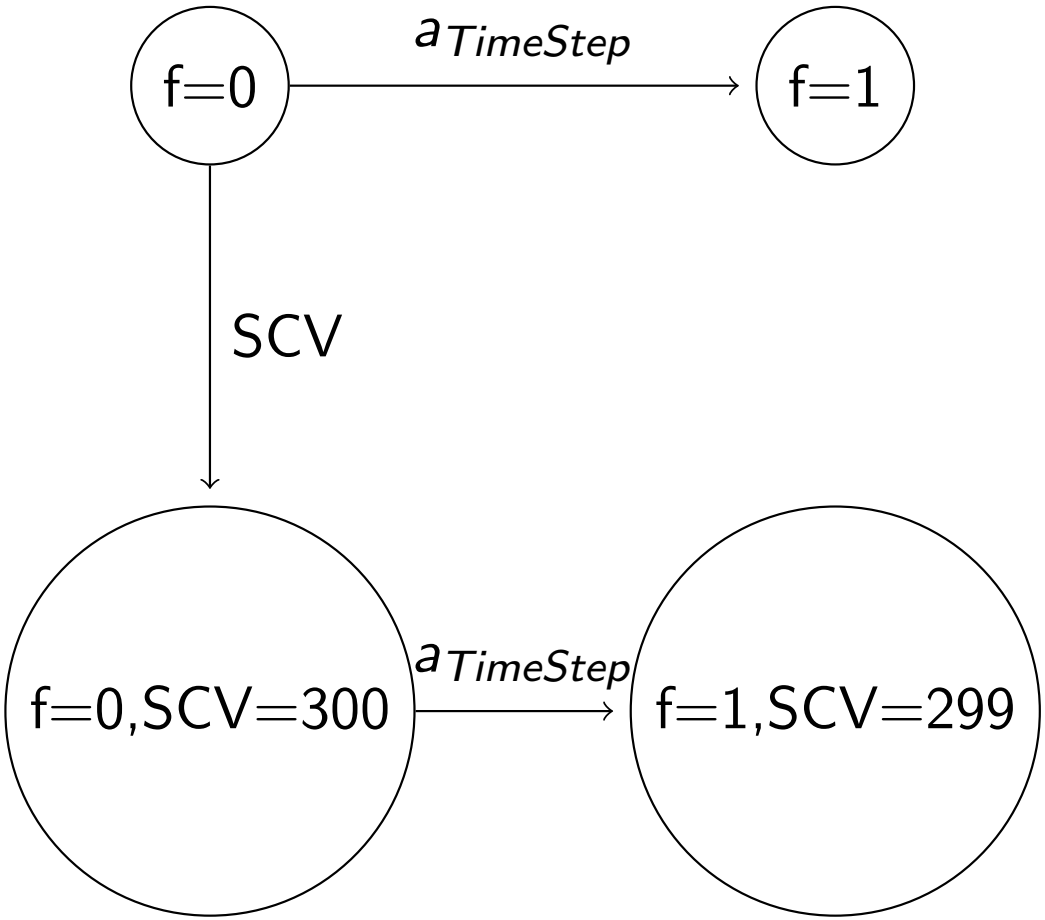
Initial State as Example

$$s_0 = (0, \{(Terran\_SCV, \emptyset, (IDLE, \infty), \infty, 4),$$
$$(Terran\_Command\_Center, \emptyset, (IDLE, \infty), \infty, 1)\}$$
$$, \{\}, 50.0, 0.0)$$

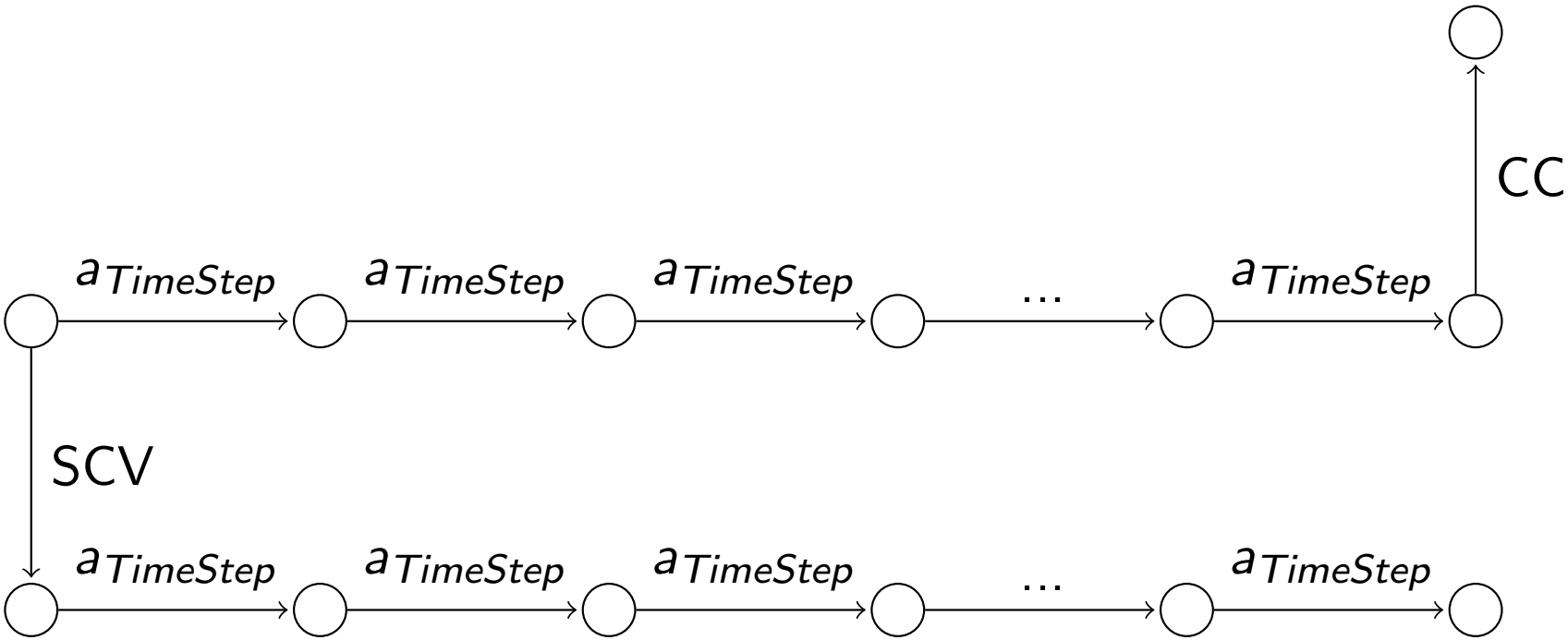
Simplifications by Churchill and Buro

- Do not consider positions.
- Worker (SCV) always collect minerals instead of being idle.
- Replace resource collecting with average income per frame.
- Combat is not part of Build Order.
- Do not cancel.
- Build as soon as possible, enables Fast Forward Mechanism.

Graph without Fast Forward Mechanism



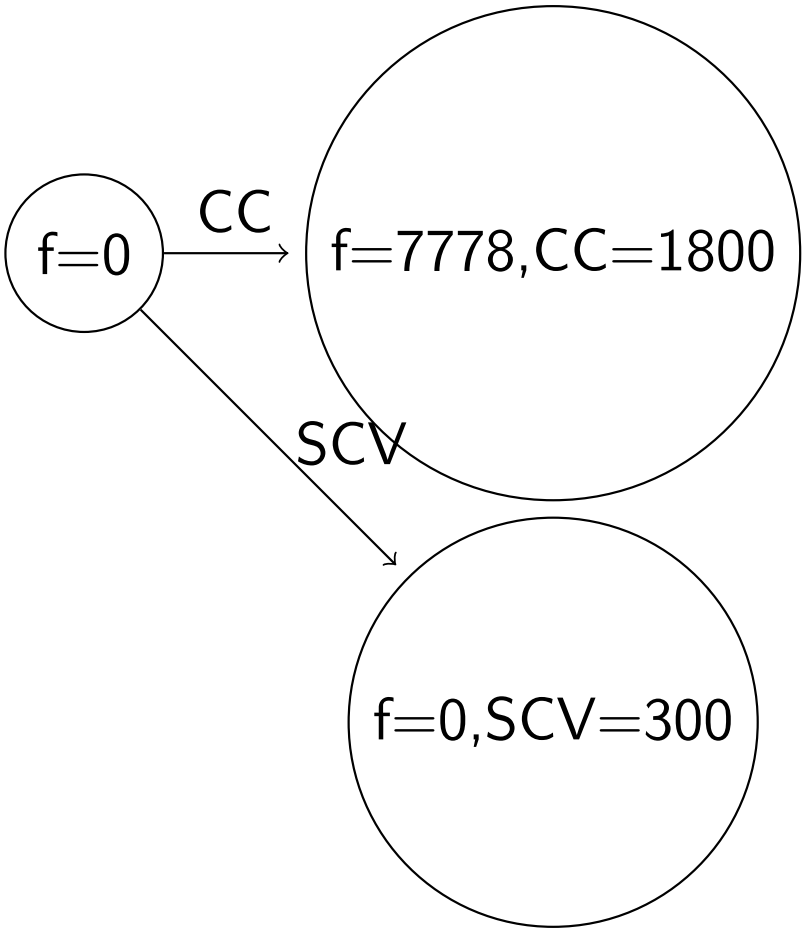
Graph without Fast Forward Mechanism



Fast Forward Mechanism

Idea: fast forward to the frame in which the unit can be build. What unit will the agent eventually be able to build when only taking $a_{TimeStep}$.

Graph with Fast Forward Mechanism



For building a Command Center, we save 7778 times the action $a_{TimeStep}$.

Action

An Action a is a 2-tuple $a := \langle o, t \rangle$

Action

An Action a is a 2-tuple $a := \langle o, t \rangle$
The number $t \in \mathbb{N}$ says by how many frames the action will fast forward.

Action possiblity 2 - without complex formula probably better?

An Action a is a 2-tuple $a := \langle o, t \rangle$
The number $t \in \mathbb{N}$ says by how many frames the action will fast forward.
The component o is contains the conditions and effects of the temporal action for building a unit.

Action Example

The action for building a CC in the initial state is

$$a_{CC\_I} = \langle((\{(\mathit{Terran\_SCV}, \mathit{NOPARTNER}, 1)\}, \emptyset), \{(\mathit{Terran\_SCV}, \mathit{NOPARTNER}, 1)\}, (400, 0, \emptyset), 1896, \emptyset, \mathit{Terran\_Command\_Center}), 7778\rangle$$

Build Order

A Build Order is a solution path in our formalism.
Example: starting in the initial state with the goal
$2 \times Terran\_Command\_Center$.
Most trivial Build Order BO would be: $BO = (a_{CC\_I})$ with
$a_{CC\_I} = \langle o_{CC}, 7778 \rangle$.

Build Order

Given an initial state and o_{CC} we have $t = 7778$ deterministically given. Therefore t is not important when talking about Build Order.
Furthermore, there exists only one o_X for every type of unit X.
We can write a Build Order just as the sequence of unit types:
$BO = (Terran\_Command\_Center)$

Make Span and Finishing Step

The make span is the duration of the whole Build Order.
Just adding up the durations of the actions would be incomplete.
Additionally we need a finishing step to advance the amount of frames the longest temporal action still needs to end.
In our example $BO = (Terran\_Command\_Center)$, the finishing step fast forwards by 1896 frames.
So the overall make span of BO is 9674 frames.

DEMO

Discussion

- The formalism allows for Build Order search for StarCraft Brood War.
- Can also be applied to other RTS games.
- Cannot handle all RTS games, for example in Age of Empires 2 the resources simplification will probably be very weak.
- When adaptations are required, this formalism can be used as basis.

My thanks go to Malte Helmert, Dave Churchill and Martin Wehrle.

Questions?

Thank you for listening and have a lovely afternoon.