

AFP Ants assignment

Bert Massop

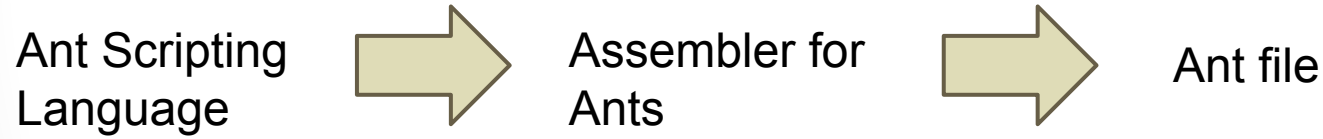
Nico Naus

Mathijs Baaijens

What we will talk about

- Ants compiler
 - Assembler for Ants (AFA)
 - Ants scripting language (ASL)
 - IDE
- Ants strategy
 - Collector
 - Defender
- Simple IDE

Multiple compiler stages

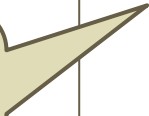


Assembler For Ants

AFA, a simple Ant assembler language.

Features

- All native instructions
- Linear program flow by default
- Labels ☺
- JUMP / GOTO / NOP



```
Sense Here 5 1 Home  
Move 2 2  
Flip 2 3 4  
Turn Left 0  
Turn Right 0  
Flip 1 5 5
```

```
find-home:  
    SENSE HERE HOME OR GOTO home-move  
    GOTO found-home  
  
home-move:  
    MOVE OR JUMP 1           ; FLIP ...  
    FLIP 2 OR JUMP 3         ; TURN RIGHT  
    TURN LEFT  
    JUMP 2                   ; GOTO find-home  
    TURN RIGHT  
    GOTO find-home  
  
found-home:  
    NOP  
    GOTO found-home
```

Ant scripting language

Imperative

Java'ish (not completely)

Compiled to AFA

Lexer based on Alex (lex for Haskell)

Parser based on Happy (yacc for Haskell)

While

ASL



AFA



Ant

```
while (Move) {  
    Mark 0  
}
```

```
_1_WHILE:  
    MOVE OR GOTO _1_IFNOT  
    MARK 0  
    GOTO _1_WHILE  
_1_IFNOT:
```

```
Move 1 0  
Mark 0 0
```

If

ASL



AFA



Ant

```
if (Move) {  
    Unmark 0  
} else {  
    Mark 0  
}  
Nop
```

```
MOVE OR GOTO _1_ELSE  
UNMARK 0  
GOTO _1_END  
_1_ELSE:  
    MARK 0  
_1_END:  
    NOP
```

```
Move 1 2  
Unmark 0 3  
Mark 0 3  
Flip 1 0 0
```

Functions

ASL



AFA



Ant

```
function main() {  
    a(2)  
}  
function a(i) {  
    Mark 3  
}
```

```
START:  
    MARK 3  
    GOTO START
```

```
Mark 3 0
```


Times

ASL



AFA



Ant

```
times (i, 3) {  
    Mark i  
}
```

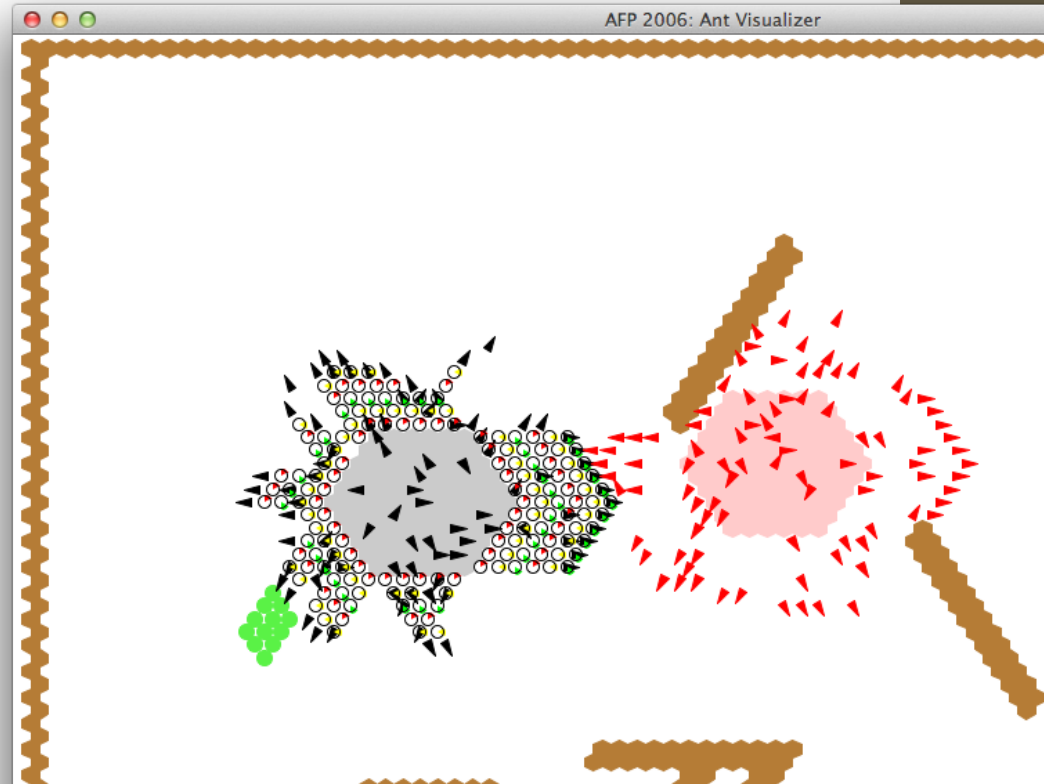
```
MARK 1  
MARK 2  
MARK 3
```

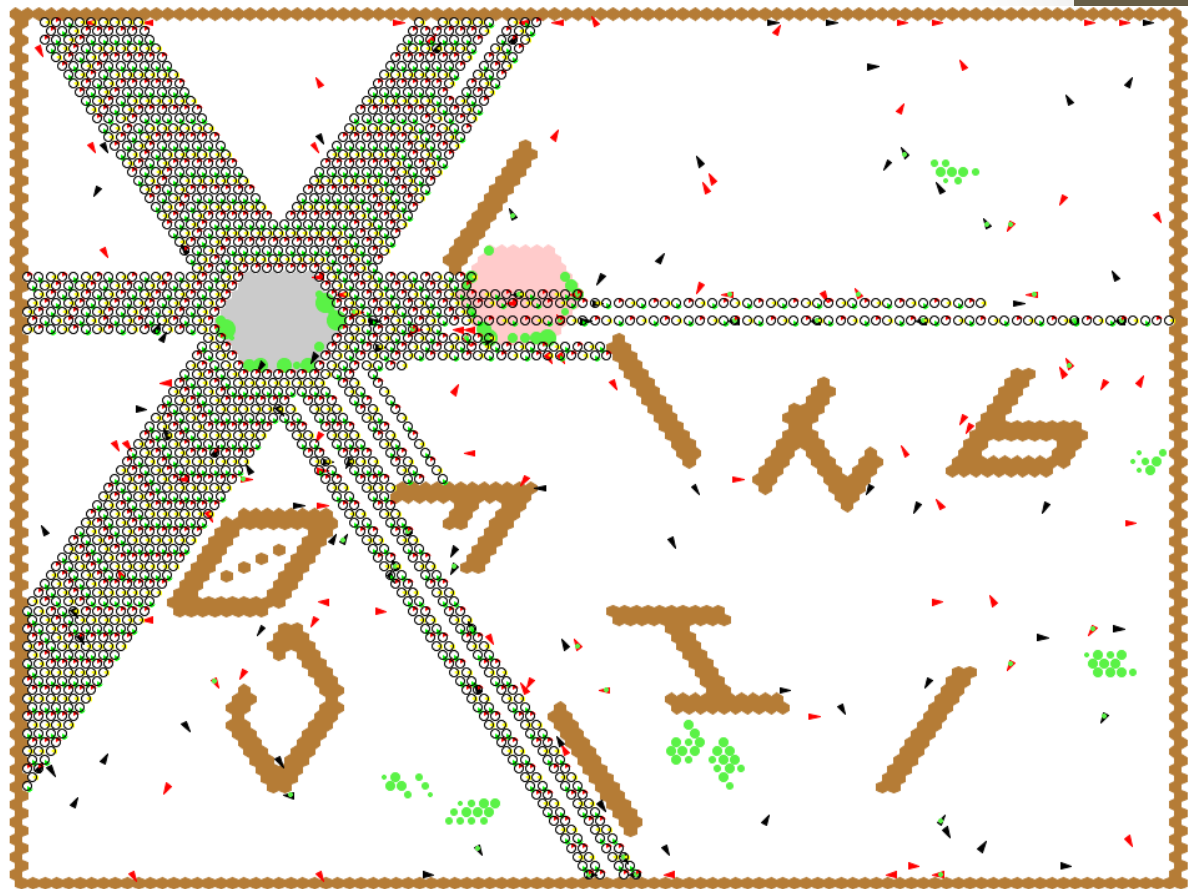
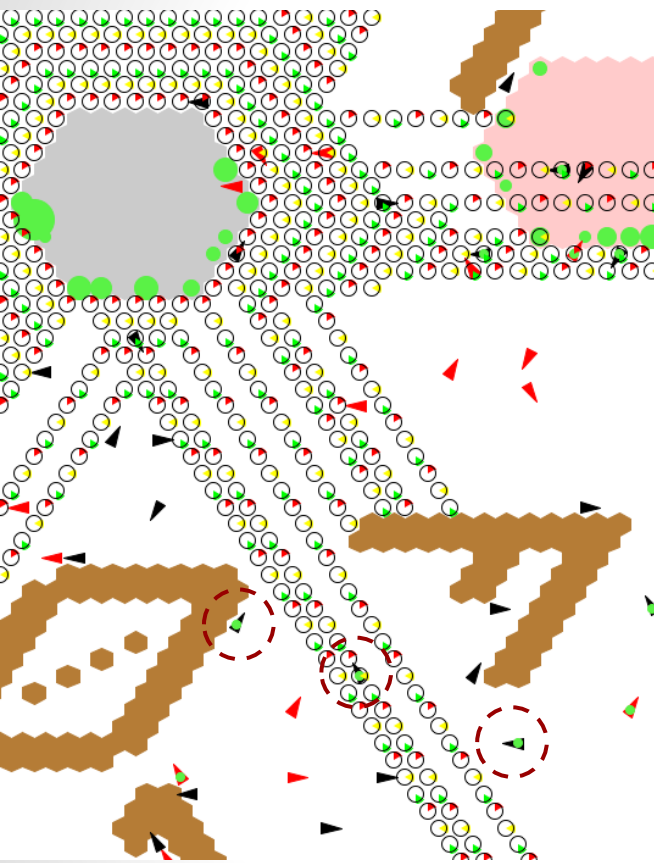
```
Mark 1 1  
Mark 2 2  
Mark 3 0
```

Ants strategy

Collector-ant

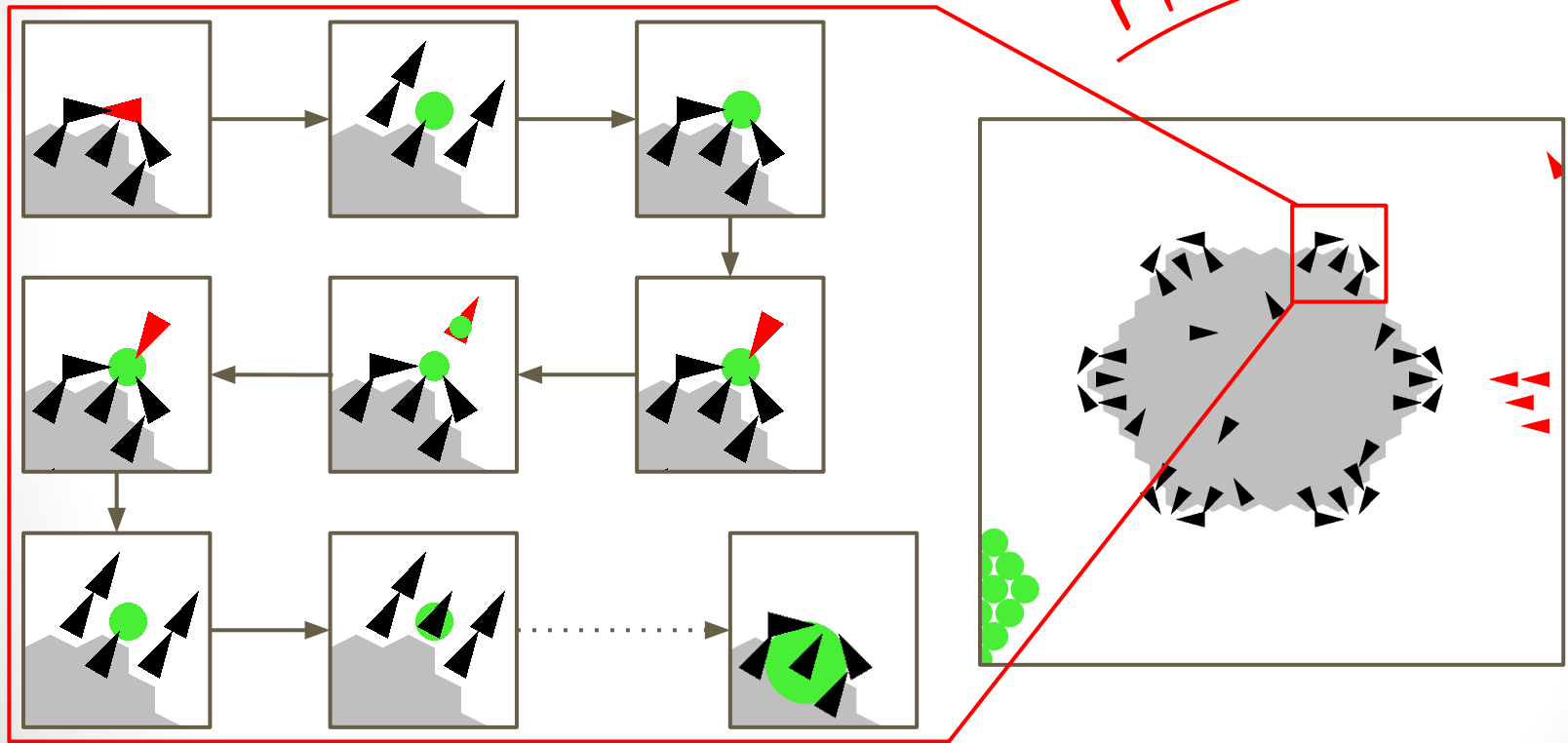
- Mark way back
- Find food
- Return home



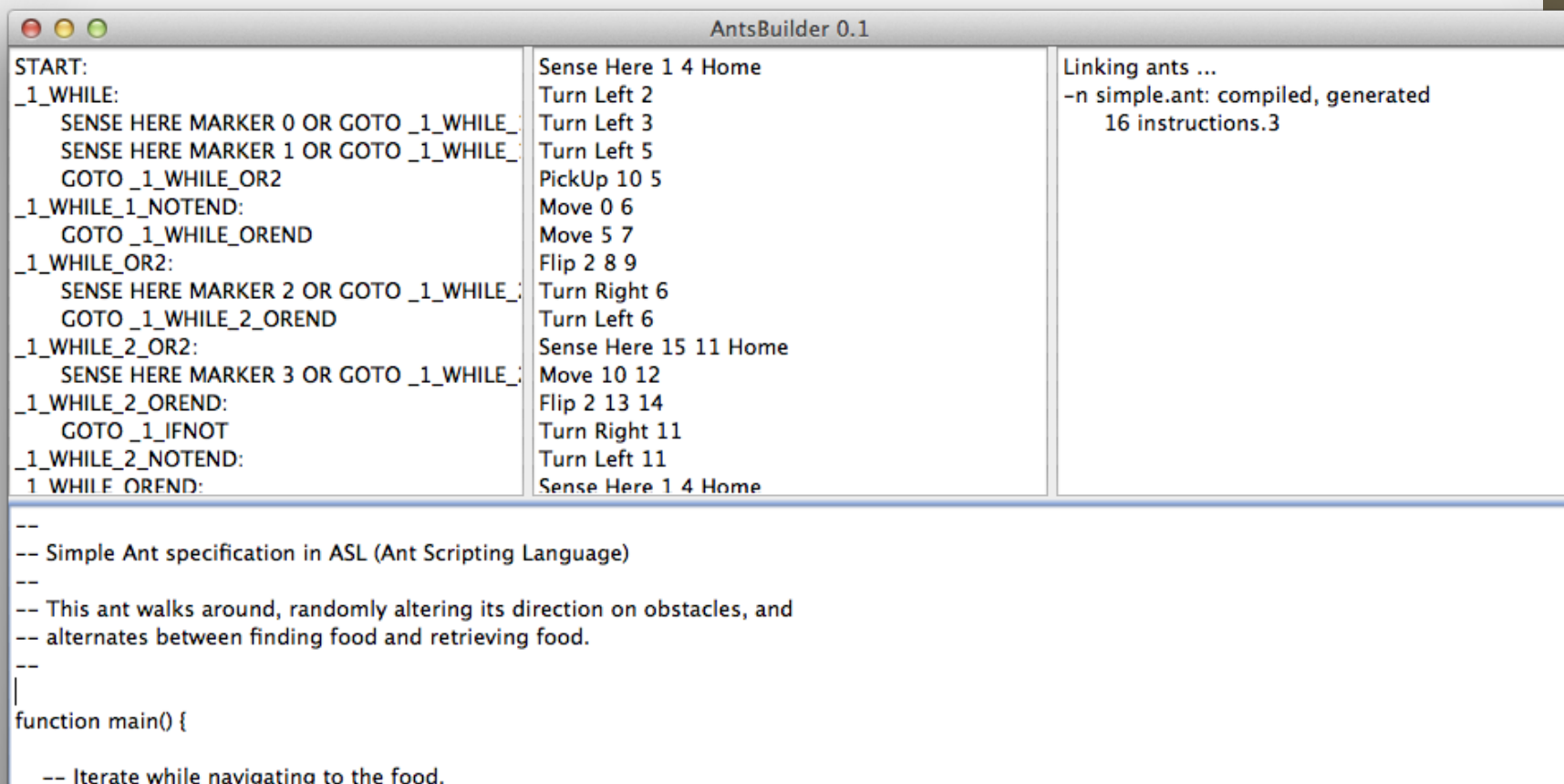


"Defending" the hill

Free food!



IDE



The screenshot shows a window titled "AntsBuilder 0.1" with three panes. The left pane contains the source ASL code, the middle pane shows the generated instructions, and the right pane displays a status message.

```
START:
_1_WHILE:
  SENSE HERE MARKER 0 OR GOTO _1_WHILE_
  SENSE HERE MARKER 1 OR GOTO _1_WHILE_
  GOTO _1_WHILE_OR2
_1_WHILE_1_NOTEND:
  GOTO _1_WHILE_OREND
_1_WHILE_OR2:
  SENSE HERE MARKER 2 OR GOTO _1_WHILE_
  GOTO _1_WHILE_2_OREND
_1_WHILE_2_OR2:
  SENSE HERE MARKER 3 OR GOTO _1_WHILE_
_1_WHILE_2_OREND:
  GOTO _1_IFNOT
_1_WHILE_2_NOTEND:
_1_WHILE_OREND:
```

```
Sense Here 1 4 Home
Turn Left 2
Turn Left 3
Turn Left 5
PickUp 10 5
Move 0 6
Move 5 7
Flip 2 8 9
Turn Right 6
Turn Left 6
Sense Here 15 11 Home
Move 10 12
Flip 2 13 14
Turn Right 11
Turn Left 11
Sense Here 1 4 Home
```

Linking ants ...
-n simple.ant: compiled, generated
16 instructions.3

```
--
-- Simple Ant specification in ASL (Ant Scripting Language)
--
-- This ant walks around, randomly altering its direction on obstacles, and
-- alternates between finding food and retrieving food.
--
|
function main() {
  -- Iterate while navigating to the food.
```