Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"

<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Using grammar chromosome: `5 16 22 29 8 10 18 7 15 11 23` we get
  - `5 mod 2 = 1: <def_fun_u> → <def_fun_u> <def_fun_s>`
  - `16 mod 2 = 0: <def_fun_u> → <def_fun_s>`
    `<def_fun_s> <def_fun_s>`

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"
<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Using grammar chromosome: `5 16 22 29 8 10 18 7 15 11 23` we get
  - 5 mod 2 = 1: <def_fun_u> → <def_fun_u> <def_fun_s>
  - 16 mod 2 = 0: <def_fun_u> → <def_fun_s>
    <def_fun_s> <def_fun_s>
    <def_fun_s> → "public void adf*(){" <adfcode> "}"
  - 22 mod 2 = 0: <adfcode> → <adfline>
  - 29 mod 2 = 1: <adfline> → <adfop>
  - 8 mod 3 = 2: <adfop> → move();

    "public void adf*() { move(); }" <def_fun_s> // first adf*()

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"

<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Using grammar chromosome: 5 16 22 29 8 10 18 7 15 11 23 we get
  - 5 mod 2 = 1: <def_fun_u> → <def_fun_u> <def_fun_s>
  - 16 mod 2 = 0: <def_fun_u> → <def_fun_s>
    <def_fun_s> <def_fun_s>
    <def_fun_s> → "public void adf*(){" <adfcode> "}"
  - 22 mod 2 = 0: <adfcode> → <adfline>
  - 29 mod 2 = 1: <adfline> → <adfop>
  - 8 mod 3 = 2: <adfop> → move();
    "public void adf*() { move(); }" <def_fun_s> // first adf*()
    <def_fun_s> → "public void adf*(){" <adfcode> "}"
  - 10 mod 2 = 0: <adfcode> → <adfline>
  - 18 mod 2 = 0: <adfline> → <adfcondition>
    <adfcondition> → if (food_ahead()==1) {<adfline>} else {<adfline>}
  - 7 mod 2 = 1: <adfline> → <adfop>
  - 15 mod 3 = 0: <adfop> → left();
  - 11 mod 2 = 1: <adfline> → <adfop>
  - 22 mod 3 = 0: <adfop> → right();
    "public void adf*(){ if (food_ahead()==1) {left();} else {right();} }" // second adf*()

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"

<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Grammar chromosome: `5 16 22 29 8 10 18 7 15 11 23`

  Final solution grammar is

```
<prog>       ::= …
    …
<op>         ::= …
public void adf*() { move(); }
public void adf*(){ if (food_ahead()==1) { left(); } else { right(); } }
```

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"
<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Grammar chromosome: `5 16 22 29 8 10 18 7 15 11 23`

  Final solution grammar is

```
<prog>       ::= …
      …
<op>         ::= …
public void adf*() { move(); }
public void adf*(){ if (food_ahead()==1) { left(); } else { right(); } }
```

- Using solution chromosome `17 24 7 15 21 1 14 15 29` we get

  `public Test(){ while(get_Energy_Left()) { <code> } }`

  - `17 mod 2 = 1: <code> → <code> <line>`
  - `24 mod 2 = 0: <code> → <line>`

  `<line> <line>`

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"

<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Grammar chromosome: `5 16 22 29 8 10 18 7 15 11 23`

  Final solution grammar is

```
<prog>       ::= …
     …
<op>         ::= …
public void adf*() { move(); }
public void adf*(){ if (food_ahead()==1) { left(); } else { right(); } }
```

- Using solution chromosome `17 24 7 15 21 1 14 15 29` we get

  `public Test(){ while(get_Energy_Left()) { <code> } }`

  - `17 mod 2 = 1: <code> → <code> <line>`
  - `24 mod 2 = 0: <code> → <line>`

    `<line> <line>`
  - `7 mod 2 = 1: <line> → <op>`
  - `15 mod 4 = 3: <op> → adf*()`
  - `21 mod 2 = 1: adf*() → **if (food_ahead()==1) { left(); } else { right(); }**`

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move();  adf*();"
<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- Grammar chromosome: `5 16 22 29 8 10 18 7 15 11 23`

  Final solution grammar is

```
<prog>      ::= …
    …
<op>        ::= …
public void adf*() { move(); }
public void adf*(){ if (food_ahead()==1) { left(); } else { right(); } }
```

- Using solution chromosome `17 24 7 15 21 1 14 15 29` we get

  `public Test(){ while(get_Energy_Left()) { <code> } }`

  - `17 mod 2 = 1: <code> → <code> <line>`
  - `24 mod 2 = 0: <code> → <line>`

    `<line> <line>`
  - `7 mod 2 = 1: <line> → <op>`
  - `15 mod 4 = 3: <op> → adf*()`
  - `21 mod 2 = 1: adf*() → if (food_ahead()==1) { left(); } else { right(); }`
  - `1 mod 2 = 1: <line> → <op>`
  - `14 mod 4 = 2: <op> → move();`

Grammatical Evolution by GE: Example

- Meta grammar for the artificial ant problem - quotes are used to escape symbols, e.g. not expand non-terminals in the meta-grammar, instead expand them in the solution grammar.

```
adf_mg - A meta-grammar that can evolve grammars.
<g> ::=
    <def_fun_u>
    "<prog>      ::= public Test() { while(get_Energy_Left()) { <code>} } "
    "<code>      ::= <line> | <code> <line>"
    "<line>      ::= <condition> | <op>"
    "<condition> ::= if (food_ahead()==1) { <line> } else { <line>}"
    "<op>        ::=  left(); | right(); | move(); | adf*();"

<def_fun_u>    ::= <def_fun_s> | <def_fun_u> <def_fun_s>
<def_fun_s>    ::= "public void adf*() {" <adfcode> "}"
<adfcode>      ::= <adfline> | <adfcode> <adfline>
<adfline>      ::= <adfcondition> | <adfop>
<adfcondition> ::= if (food_ahead()==1) { <adfline> } else { <adfline> }
<adfop>        ::= left(); | right(); | move();
```

Main program

ADFs' definitions

- **Final executable code** for

  grammar chromosome: 5 16 22 29 8 10 18 7 15 11 23

  and

  solution chromosome: 17 24 7 15 18 1 14 15 29

  is

```
public Test(){
   while(get_Energy_Left()) {
      if (food_ahead()==1) {
         left();
      } else {
         right();
      }
      move();
   }
}
```

first adf*()