Pool evolution Pattern

P population {set of individuals}

S selection function Population → Population

E evolution function Individual → Individual

F filter function Population → Population
(look at the population and select just some others)

t termination function Population → boolean

the evolution pattern can be expressed like this:

```
while(¬t(L(P)))
    N:= E(S(P))
    P:= P ∪ {N(P, P)}
}
```

filtering of new individuals with respect to the current population

(6.13)

MEMORY: $P_0 \to P_1 \to P_2 \to \dots \to P_n$

$T \cdot n \cdot Type$

Well we can cheat do:
→ Brand can't compute to whole another special compute to become those is a dependency between the iterations.

If I have the possibility to speed up (6.14)

$P_n \to P_{n+1}$

time
Type

$Sp = \frac{time}{Tpar}$

Well Can we do to parallelize each single iteration?

```
while(¬t(L(P)))
    N:= E(S(P))
    P:= P ∪ {N(P, P)}
}
```

} parallelizes.

$N := E(S(P))$

(6.15)

- Speed the selection pattern!
  - this is parallel to be and depends on the function that I use to split.

$P' \cup P'' = P$

$S(P') \parallel S(P'')$

$\cup$

range

If the selection function allows me to do this the selection function is easily parallelizable.

If I have the property:

After the selection I have to apply the evolution function.

S
P
P
P