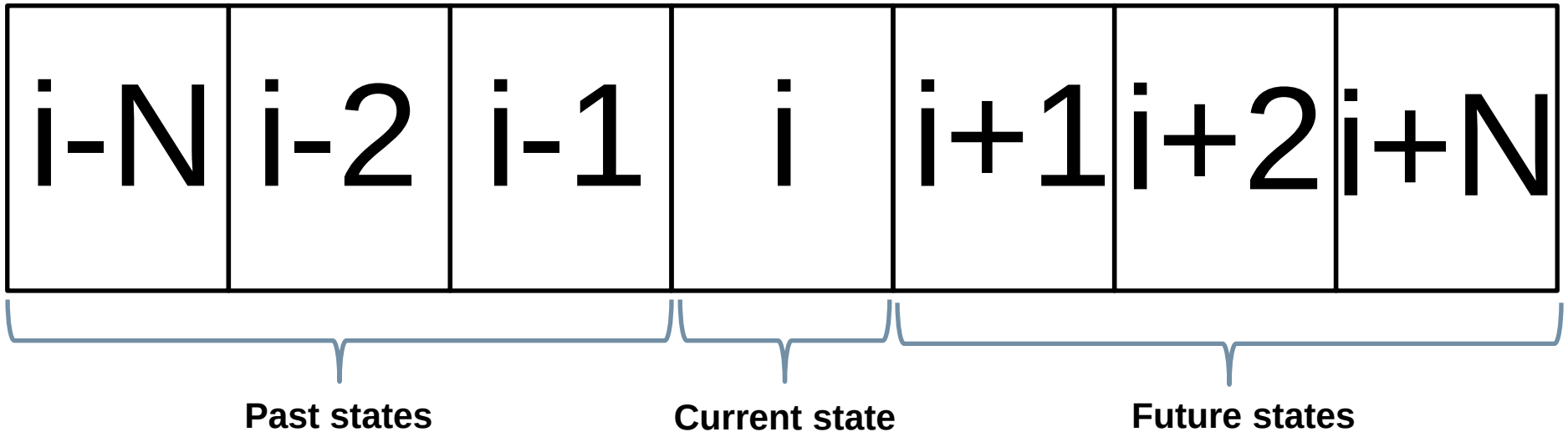
ALLOY 6-Syntactic overview

TEMPORAL CONNECTIVES

Authors:
Luca Padalino
Francesca Pia Panaccione
Francesco Santambrogio

LINEAR TEMPORAL LOGIC (LTL)
Definition

LINEAR TEMPORAL LOGIC (LTL): *«an infinite sequence of states where each point in time has a unique successor, based on a linear-time perspective»*[1]



[1] Ashari, R., & Habib, S. (n.d.). *LINEAR TEMPORAL LOGIC (LTL)*.

SIGNATURES

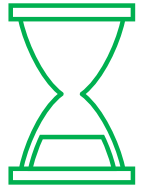
```
sig Person {
var liveness: Liveness
}

enum Liveness {Alive, Dead}
```

PREDICATES

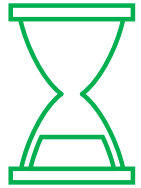
```
pred Die [p: Person] {
p.liveness = Alive
p.liveness' = Dead }
```

To express constraints that hold at different instants of time or for a certain amount of time



- How can we express that a person is not immortal?
- How can we express that a person cannot come back to life?

To express constraints that hold at different instants of time or for a certain amount of time



- How can we express that a person is not immortal?
- How can we express that a person cannot come back to life?

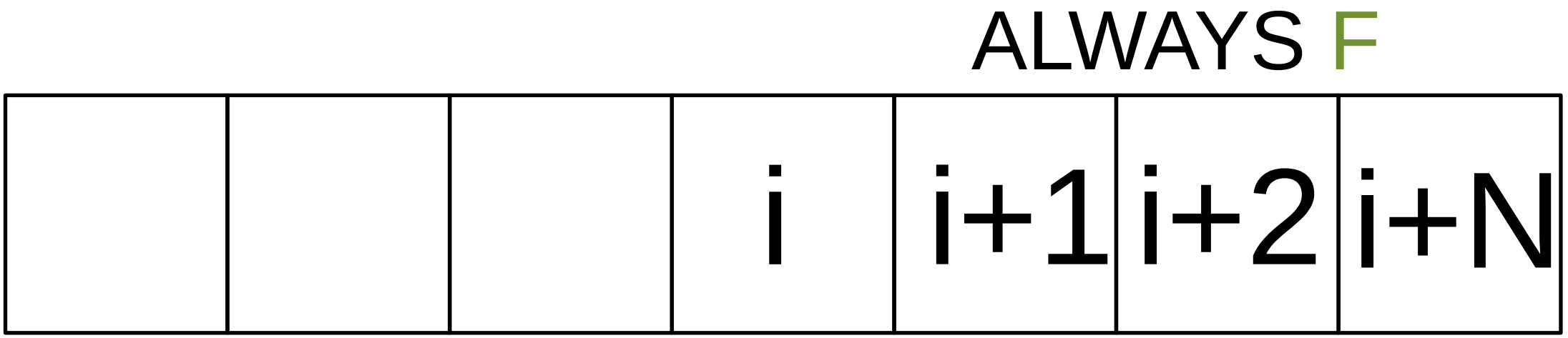
TEMPORAL CONNECTIVES

Sintax: **Exp::= un0p expr| expr bin0p expr**

F IS TRUE

- **Un0p::= always| after| eventually**
- **Bin0p::= until| releases| ;**

ALWAYS F = true in i **iff** F = true in k>=i
for each state k



ALWAYS

```
fact NoResurrection {
always (all p:Person|
p.liveness = Dead
implies always
p.liveness = Dead)}
```

TEMPORAL CONNECTIVES
After

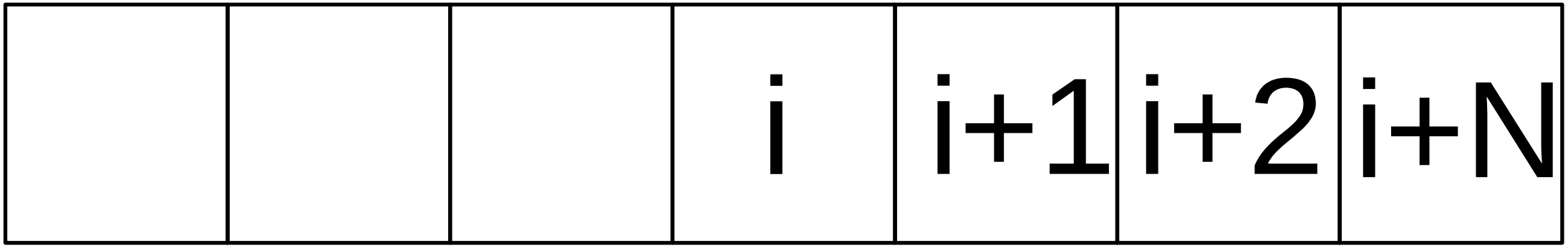
Sintax: **Exp::= un0p expr| expr bin0p expr**

F IS TRUE

- **Un0p::= always| after| eventually**
- **Bin0p::= until| releases| ;**

AFTER F = true in i **iff** F = true in i+1

AFTER S = S'



AFTER

```
assert NoResurrection {
always (all p:Person|
p.liveness = Dead
implies after
p.liveness = Dead)}
```

TEMPORAL CONNECTIVES
Eventually

Sintax: **Exp::= un0p expr| expr bin0p expr**

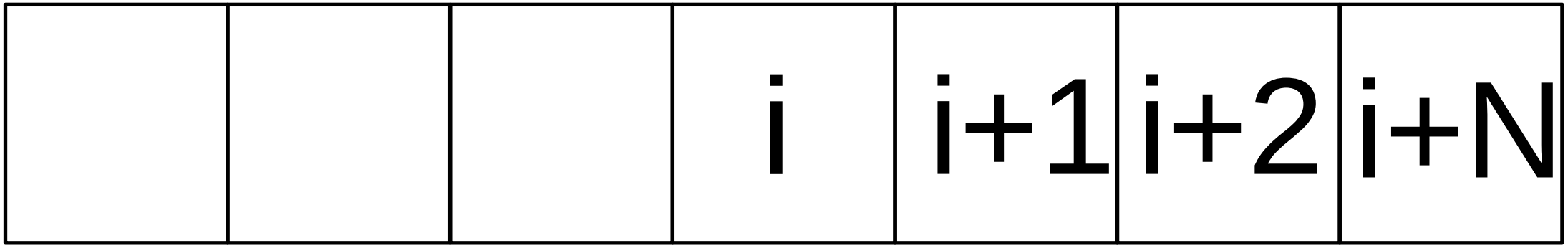
F IS TRUE

- **Un0p::= always| after| eventually**
- **Bin0p::= until| releases| ;**

EVENTUALLY F = true in i **iff** F = true in k>=i
for some state k

LIVENESS PROPERTY

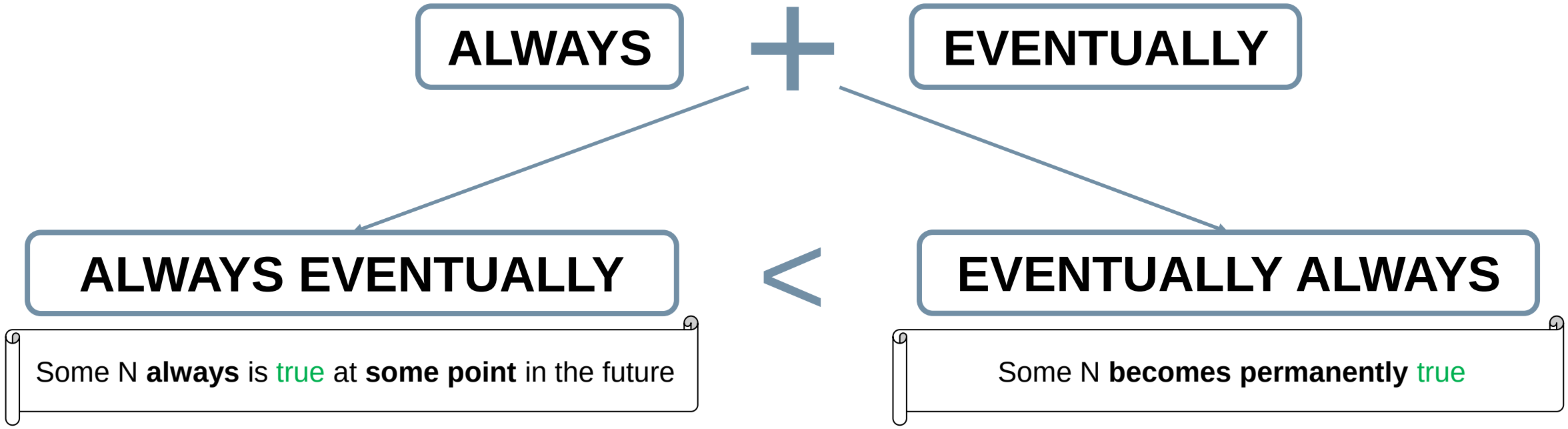
EVENTUALLY F



EVENTUALLY

```
fact noImmortality {
always (all p:Person |
p.liveness = Alive implies
after (eventually
p.liveness = Dead))}
}
```

TEMPORAL CONNECTIVES
Combined expression



ALWAYS
+
EVENTUALLY

```
pred Mortality1 {
all p:Person |
always eventually
p.liveness = Dead
}
pred Mortality2 {
all p:Person |
eventually always
p.liveness = Dead
}
```

TEMPORAL CONNECTIVES
Historically

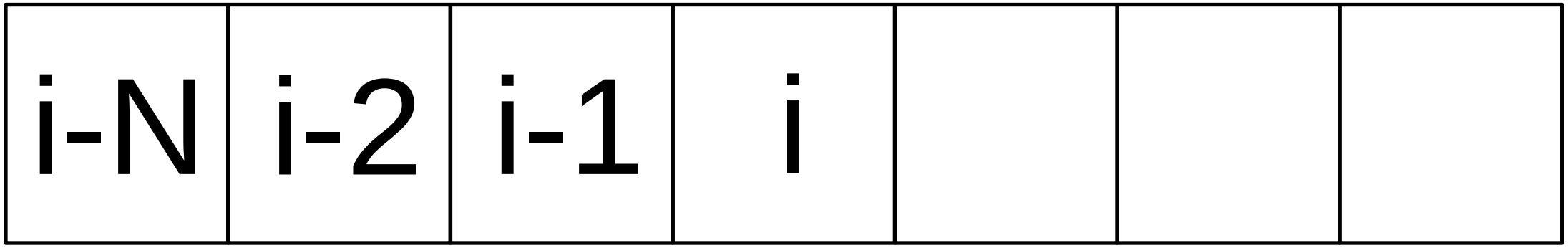
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= historically| once| before**
- **Bin0p::= since| triggered|**

● **F IS TRUE**

HISTORICALLY F = true in i **iff** F = true in k<=i
for each state k

HISTORICALLY F



HISTORICALLY

```
fact NoDeadThenAlive {
always (all p:Person |
p.liveness = Alive
implies historically
p.liveness = Alive)}
```

TEMPORAL CONNECTIVES
Once

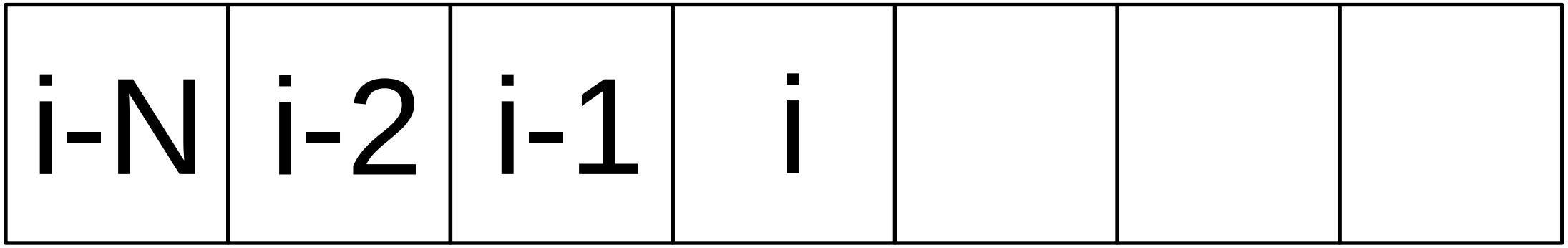
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= historically| once| before**
- **Bin0p::= since| triggered|**

● **F IS TRUE**

ONCE F = true in i **iff** F = true in k<=i
for some state k

ONCE F



ONCE

```
fact DeadSinceDeath {
always (all p:Person |
p.liveness = Dead implies
once Die [p])}
```

TEMPORAL CONNECTIVES
Before

Sintax: **Exp::= un0p expr| expr bin0p expr**

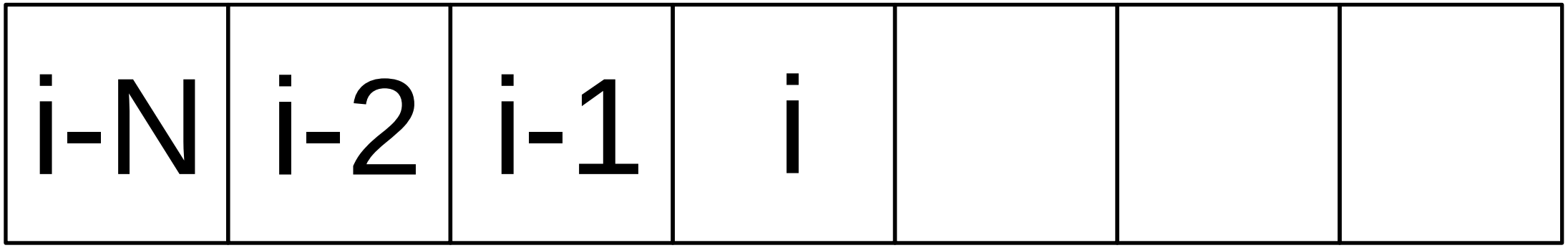
- **Un0p::= historically| once| before**
- **Bin0p::= since| triggered|**

● **F IS TRUE**

BEFORE F = true in i **iff** F = true in i-1
for i>0

BEFORE F

FALSE IN STATE 0



BEFORE

```
assert IfAliveBeforeAlive {
after
(always (all p:Person |
p.liveness = Alive implies
before p.liveness = Alive)
)}
```

TEMPORAL CONNECTIVES
Until

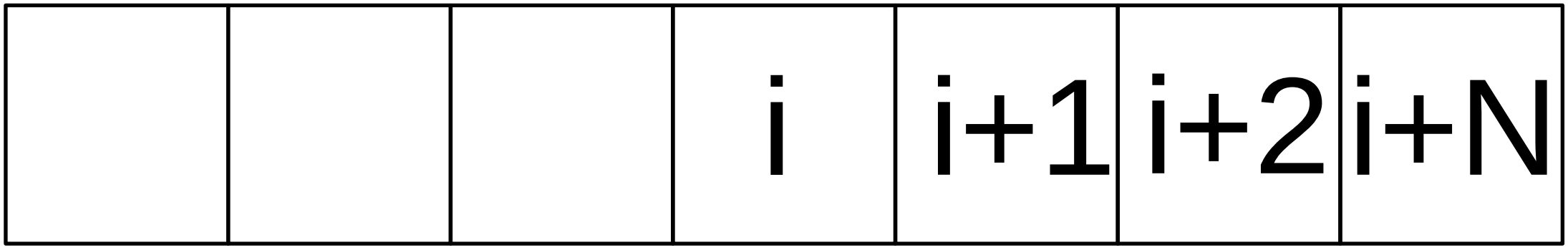
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= always| after| eventually**
- **Bin0p::= until| releases| ;**

F UNTIL G = true in i **iff** G = true in j>=i **&&** F = true in k: i<=k<j
for some state j, **for each** state k

● **F IS TRUE**
● **G IS TRUE**

F UNTIL G



UNTIL

```
fact AliveUntilDeath {
always (all p:Person |
p.liveness = Alive implies
(p.liveness=Alive until
p.liveness = Dead))}
```

TEMPORAL CONNECTIVES
Releases

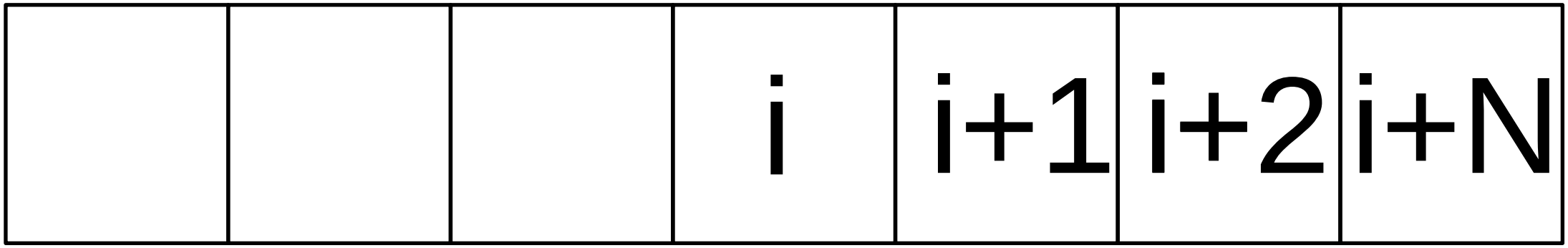
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= always| after| eventually**
- **Bin0p::= until| releases| ;**

F IS TRUE
G IS TRUE

F RELEASES G = true in i **iff** F = true in k **&&** G = true in j: i<=j<=k
for some state k, **for each** state j
OR
no such k **&&** G = true in j>=i **for each** j

F RELEASES G



RELEASES

```
assert AliveUntilDeath2 {
always (all p:Person |
p.liveness = Alive implies
(Die[p] releases
p.liveness = Alive))}
```

TEMPORAL CONNECTIVES
Triggered

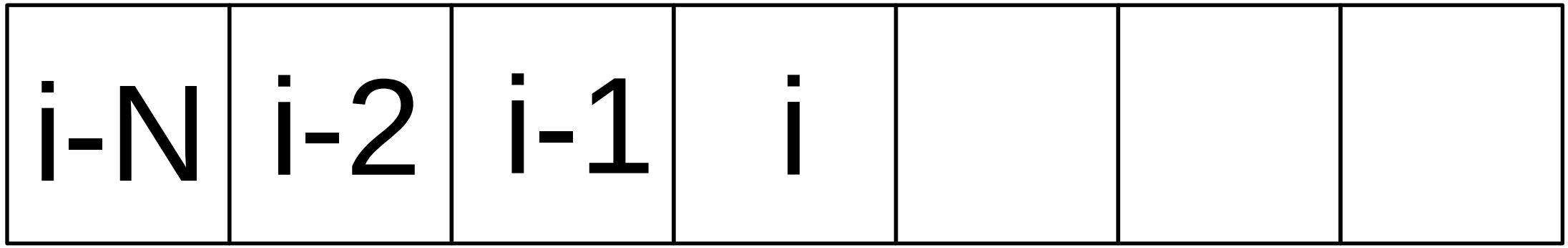
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= historically| once| before**
- **Bin0p::= since| triggered**

F IS TRUE
G IS TRUE

F TRIGGERED G = true in i **iff** F = true in j<=i **&&** G = true in k: j<k<=i
for some state j, **for each** state k
OR
F = false in j<=i **&&** G = true in k<=j
for each j, **for each** k

F TRIGGERED G



TRIGGERED

```
assert DeadSinceDeath2 {
always (all p: Person |
p.liveness = Dead implies
(Die[p] triggered
p.liveness = Dead))}
```

TEMPORAL CONNECTIVES
Since

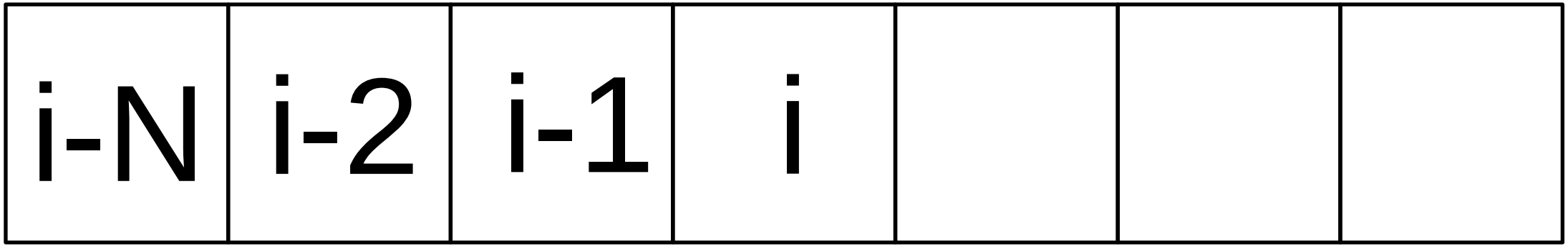
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= historically| once| before**
- **Bin0p::= since| triggered**

F IS TRUE
G IS TRUE

F ONCE G = true in i **iff** G = true in j<=i **&&** F = true in k: j<k<=i
for some state j, **for each** state k

F SINCE G



SINCE

```
assert DeadSinceDeath2 {
always (all p: Person |
p.liveness = Dead implies
(p.liveness=Dead since
Die[p]))}
```

TEMPORAL CONNECTIVES
;

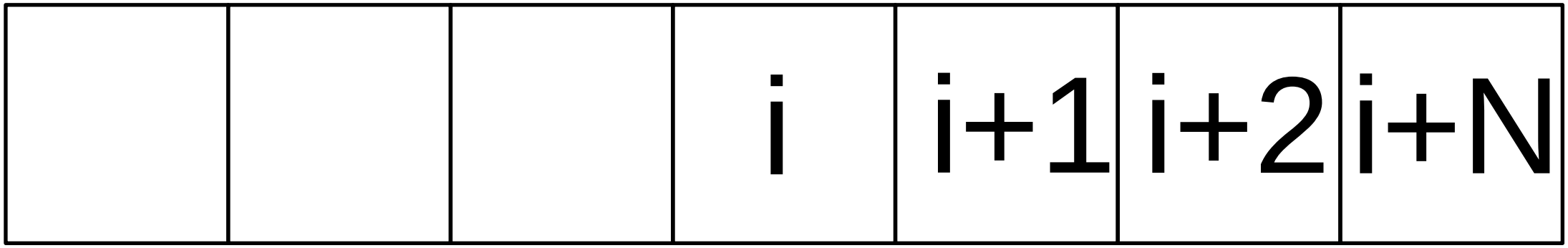
Sintax: **Exp::= un0p expr| expr bin0p expr**

- **Un0p::= always| after| eventually**
- **Bin0p::= until| releases| ;**

F ; G = true in i **iff** F true in i **&&** G = true in i+1

; = AND AFTER

F ; G



F IS TRUE

G IS TRUE

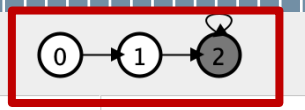
TEMPORAL CONNECTIVES
;

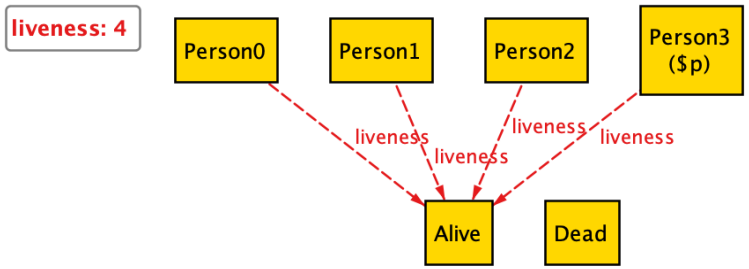
;

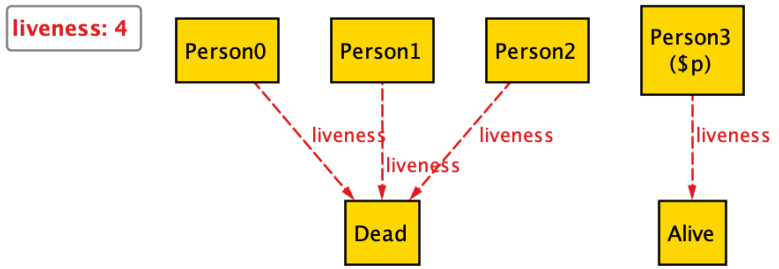
```
run{#Person = 4 and
some p: Person |
(p.liveness = Alive ;
p.liveness = Alive ;
p.liveness = Dead)}
for 5
```

TEMPORAL CONNECTIVES

;







TEMPORAL CONNECTIVES
;