# Sémantique dénotationnelle, espaces métriques et Go

GDR SciLog, GT LVP, 2025

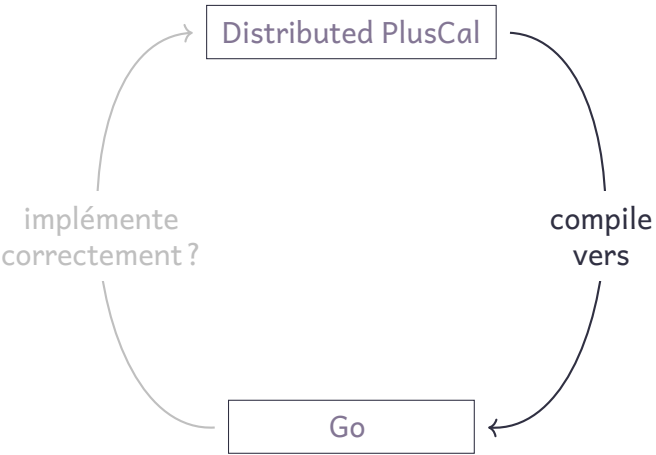
Ghilain Bergeron

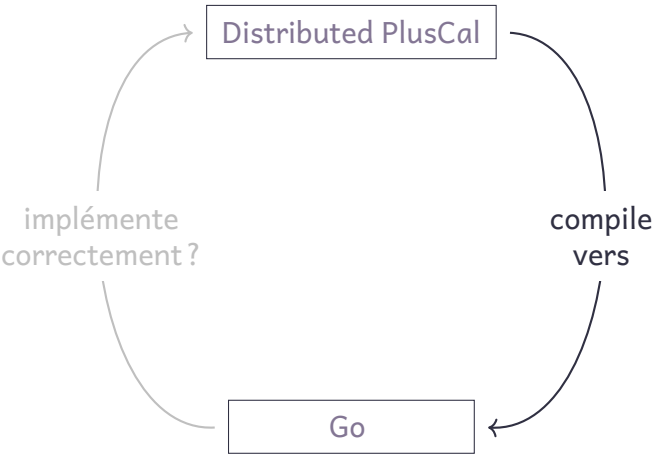
Université de Lorraine,
CNRS, Inria, LORIA

14 novembre 2025




# Il était une fois…

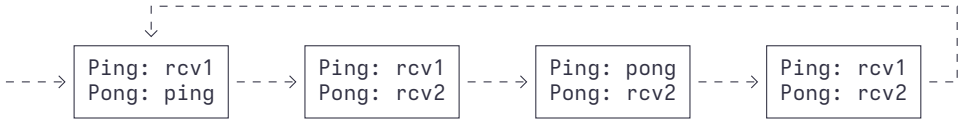



Distributed PlusCal
compile
vers
Go
implémente
correctement ?

Distributed PlusCal est un langage de spécification d'algorithmes distribués/concurrents…

* …qui est traduit en une spécification $TLA^+$.
* …avec des primitives de parallélisme (local) et de communication.
* …qui ressemble au langage C.

```
algorithm PingPong {
    fifos ping, pong;

    process (Ping = 0)
        variable tmp1 = "";
    {
    rcv1:   receive(ping, tmp1);
            await tmp1 = "Ping";
            goto pong;
    pong:   send(pong, "Pong");
            goto rcv1;
    }

    process (Pong = 1)
        variable tmp2 = "";
    {
    ping:   send(ping, "Ping");
            goto rcv2;
    rcv2:   receive(pong, tmp2);
            await tmp2 = "Pong";
            goto ping;
    }
}
```



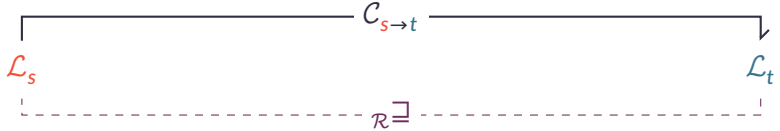
Go est un langage de programmation…

- …concurrente (coroutines, channels, …).
- …impérative (boucles, variables mutables, …).
- …relativement minimaliste.

```go
func Ping(
  ping <-chan string,
  pong chan<- string,
) {
  tmp1 := ""
  ok1 := true

  for true {
    if tmp1, ok1 = <-ping; !ok1 {
      break
    }
    if tmp1 != "Ping" {
      continue
    }
    pong <- "Pong"
  }
}
```

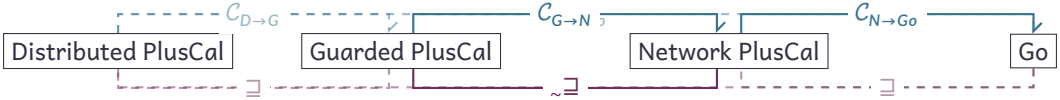
```go
func Pong(
  ping chan<- string,
  pong <-chan string,
) {
  tmp2 := ""
  ok2 := true

  for true {
    ping <- "Ping"
    if tmp2, ok2 = <-pong; !ok2 {
      break
    }
    if tmp2 != "Pong" {
      continue
    }
  }
}
```

* Compiler, oui, mais pas n'importe comment !
* Processus compliqué, mais qui vaut le coup !
* D'autant plus important pour un langage de spécification comme Distributed PlusCal !




**Définition (Raffinement)**

Pour une relation $\mathcal{R} \subseteq \mathcal{D}_s \times \mathcal{D}_t$, un comportement $\sigma_t \in [\![\mathcal{C}_{s\to t}(P_t)]\!]_t$ est $\mathcal{R}$-*justifié par* un comportement $\sigma_s \in [\![P_s]\!]_s$ (noté $\sigma_s \ {}_{\mathcal{R}}\sqsupseteq \ \sigma_t$) si $(\sigma_s, \sigma_t) \in \mathcal{R}$.




Légende :

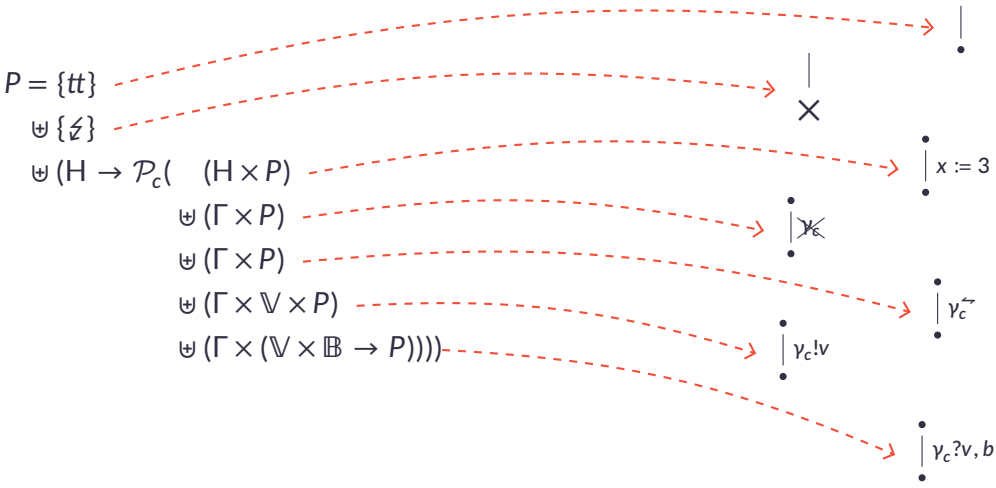
- Développé dans le langage de programmation Lean 4 (environ **4k** LoC)
- Vérifié dans l'assistant à la preuve Lean 4 (environ **9k** LoC)
- A développer ☺
- A vérifier ☺

- ✅ Sémantique dénotationnelle de Distributed PlusCal
- ❌ Sémantique dénotationnelle de Go

# Une sémantique (dénotationnelle) de Go

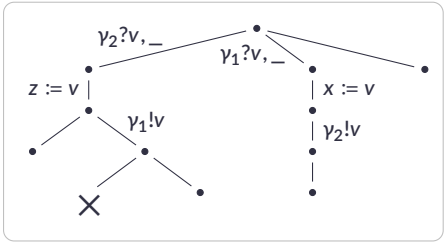
$$
\begin{aligned}
P = {} & \boxed{\{tt\}} && \text{Tout s'est bien passé} \\
& \uplus \boxed{\{\lightning\}} && \text{Crash} \\
& \uplus (\mathrm{H} \to \mathcal{P}_c( \quad \boxed{(\mathrm{H} \times P)} && \text{Changement dans la mémoire} \\
& \qquad \uplus \boxed{(\Gamma \times P)} && \text{Clôture d'un channel synchrone} \\
& \qquad \uplus \boxed{(\Gamma \times P)} && \text{Synchronisation sur un channel synchrone} \\
& \qquad \uplus \boxed{(\Gamma \times \mathbb{V} \times P)} && \text{Envoi sur un channel synchrone} \\
& \qquad \uplus \boxed{(\Gamma \times (\mathbb{V} \times \mathbb{B} \to P))})) && \text{Réception sur un channel synchrone}
\end{aligned}
$$

Cette équation a une solution : c'est notre domaine sémantique !


$P = \{tt\}$
$\uplus \{\lightning\}$
$\uplus (\mathsf{H} \to \mathcal{P}_c($ $(\mathsf{H} \times P)$
$\uplus (\Gamma \times P)$
$\uplus (\Gamma \times P)$
$\uplus (\Gamma \times \mathbb{V} \times P)$
$\uplus (\Gamma \times (\mathbb{V} \times \mathbb{B} \to P))))$
$x := 3$
$\gamma_c!v$
$\gamma_c?v, b$

```
select
| z, _ = ←γ2 ⇒
|   if P1(z) then
|   |  skip
|   else
|   |  γ1 ← z
|   |  if P2(z) then
|   |  |  panic "some error"
|   |  else
|   |  |  skip
| x, _ = ←γ1 ⇒
|   γ2 ← x
| default ⇒
|   skip
```

$$\lambda\,\mathrm{h}.\left\{\begin{array}{l}\langle\gamma_2^?, \lambda\, v\,\_\,\mathrm{h}.\,\{\langle \mathrm{h}(z := v), \lambda\,\mathrm{h}.\,\{tt, \langle\gamma_1^!, v, \lambda\,\mathrm{h}.\,\{\lightning, tt\}\rangle\}\rangle\},\\ \langle\gamma_1^?, \lambda\, v\,\_\,\mathrm{h}.\,\{\langle \mathrm{h}(x := v), \lambda\,\mathrm{h}.\,\{\langle\gamma_2^!, v, tt\rangle\}\rangle\},\\ tt\end{array}\right\}$$



*P* forme un espace métrique complet.

→ Toute suite de Cauchy dans *P* converge dans *P* !

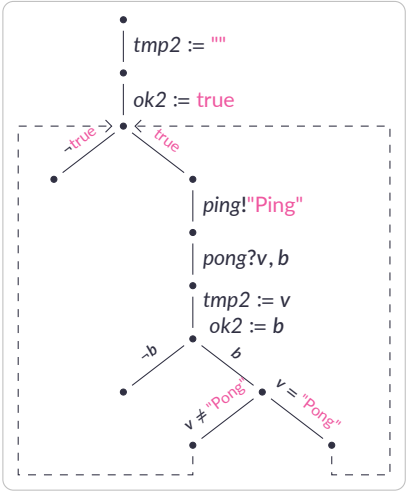
**Espaces métriques complets**

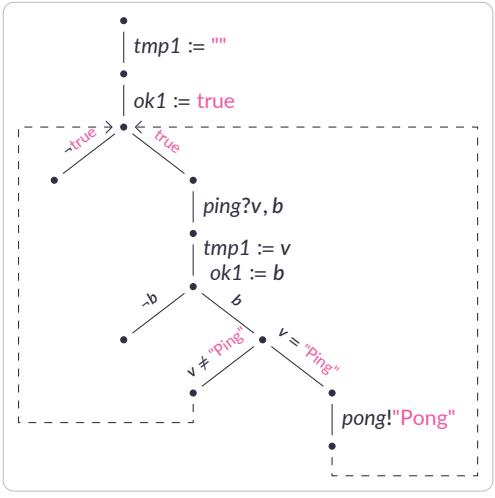
* Points limites de suites de Cauchy

**DCPO**

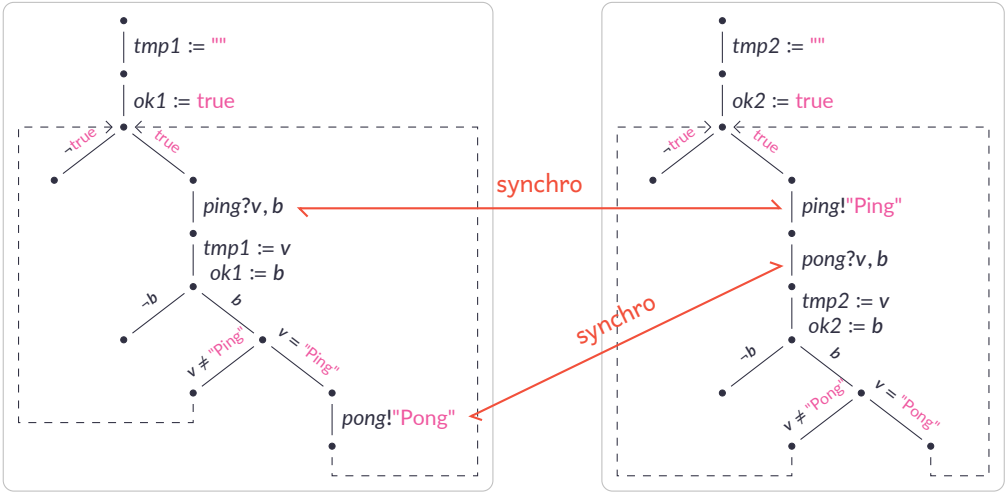
* Points fixes de fonctions continues

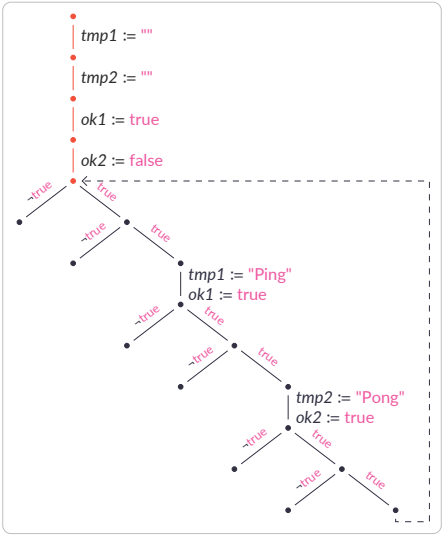
```
func Pong(
  ping chan<- string,
  pong <-chan string,
) {
  tmp2 := ""
  ok2 := true

  for true {
    ping <- "Ping"
    if tmp2, ok2 = <-pong; !ok2 {
      break
    }
    if tmp2 != "Pong" {
      continue
    }
  }
}
```



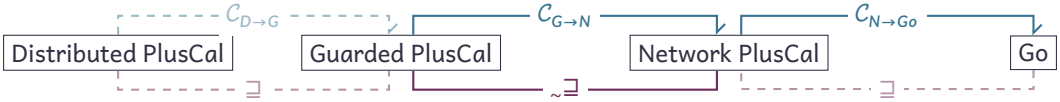



```
func Ping(
  ping <-chan string,
  pong chan<- string,
) {
  tmp1 := ""
  ok1 := true

  for true {
    if tmp1, ok1 = <-ping; !ok1 {
      break
    }
    if tmp1 != "Ping" {
      continue
    }
    pong <- "Pong"
  }
}
```


tmp1 := ""
ok1 := true
¬true
true
ping?v, b
tmp1 := v
ok1 := b
¬b
b
v ≠ "Ping"
v = "Ping"
pong!"Pong"
synchro
tmp2 := ""
ok2 := true
¬true
true
ping!"Ping"
pong?v, b
tmp2 := v
ok2 := b
¬b
b
v ≠ "Pong"
v = "Pong"
synchro


tmp1 := ""
tmp2 := ""
ok1 := true
ok2 := false
¬true
true
¬true
true
tmp1 := "Ping"
ok1 := true
¬true
true
¬true
true
tmp2 := "Pong"
ok2 := true
¬true
true
¬true
true

# Et maintenant ?




- → Preuve de correction pour $\mathcal{C}_{N\to Go}$
- → Compilateur $\mathcal{C}_{D\to G}$ ?