語法—語意—效果：程式語言的三層存在結構

Syntax, Semantics, and Effects: The Three-Layer Ontology of Programming Languages

v0.1 / 2026-07-27 / Neo.K with Aletheia / 初版完成

https://thisoneisneok.com/html/papers/pu-2-03.html

SECTION

Syntax, Semantics, and Effects: The Three-Layer Ontology of Programming Languages

論文編號： PU-2-03

系列：「程式宇宙書系」第 2 冊《程式語言的本質》

作者： Neo.K with Aletheia

機構： EveMissLab／一言諾科技有限公司

版本： v0.1

日期： 2026 年 7 月 27 日

SECTION

摘要

前兩篇分別提出：程式語言不等於文字，文字只是程式結構的一種投影；符號也不等於靜態字元，而是具有輸入、輸出、前後條件、效果、授權、驗證與組合規則的受治理算子。本篇進一步建立程式語言的三層存在結構：

> 【mathcal L
> =
> <
> mathcal Y,
> mathcal M,

其中：

- mathcal Y：Syntax，語法層；
- mathcal M：Semantics，語意層；
- mathcal E：Effects，效果層。

本文核心命題是：

> 【Syntax
> ≠
> Semantics
> ≠
> Effects】

語法回答「哪些結構可以形成」；語意回答「該結構在問題世界中代表什麼」；效果回答「當結構被執行時，實際改變了哪些數位、制度或物理世界狀態」。

本文將語法定義為合法結構形成空間：

> mathcal Y
> =
> <
> Σ,
> G,
> A,
> B
> >

其中 Σ 為符號集合，G 為形成規則，A 為抽象結構，B 為局部邊界與作用域規則。語法合法只表示某一結構能被語言接受，不表示其名稱已解析、世界關係成立、權限合法或效果可承擔。

語意則被定義為從語法結構到問題世界概念、狀態與規則的映射：

> llbracket y rrbracket_(Γ,W)
> =
> m

其中：

- y：語法結構；
- Γ：型別、作用域、身份、權限與上下文環境；
- W：問題世界；
- m：語意結構。

  同一語法在不同上下文與問題世界下，可能具有不同語意：

> llbracket y rrbracket_(Γ⊠,W⊠)
> ≠
> llbracket y rrbracket_(Γ⊠,W⊠)

因此，語法名稱不是語意本身。

效果則被定義為語意結構在 Runtime 中造成的可觀測世界差分：

> 【mathcal E(m,R,S⊠)
> =
> <
> S_(t+1),
> Events,
> ExternalEffects,
> Evidence,
> Residual
> >】

其中：

- R：Runtime 與執行環境；
- S⊠：執行前狀態；
- S_(t+1)：執行後權威狀態；
- Events：形成的事件；

- ExternalEffects：外部數位或物理效果；
- Evidence：可驗證證據；
- Residual：未完成、不確定、不可逆或補償殘差。

  本文提出三層合法性：

- 語法合法性：結構可被形成與解析；
- 語意合法性：結構指向有效世界身份、規則與狀態；
- 效果合法性：實際世界差分在權限、責任、風險與證據上可接受。

  形式上：

> 【Valid_(mathcal Y)
> not⇒
> Valid_(mathcal M)
> not⇒
> Valid_(mathcal E)】

一段程式可以語法正確、型別正確、甚至語意明確，卻因未授權、外部結果不可驗證、效果不可逆或違反治理邊界，而仍不能合法執行。

本文進一步建立「語意不完整」與「效果不完整」概念。傳統編譯器通常在語法與型別層拒絕錯誤，但對下列情形缺乏直接表示：

- 名稱指向錯誤世界實體；
- 狀態版本已過期；
- 函式具有隱藏外部效果；
- 權限只在 Runtime 外部檢查；
- 呼叫成功不代表外部效果完成；
- 部分成功與結果未知無法進入型別；
- 回滾不能撤回世界影響。

本文因此提出擴展判定：

> Γ;W;P;S
> |-
> y
> :
> T
> !
> ε
> triangleright
> Q

其中：

- T：值或結構型別；
- ε：效果簽章；
- Q：執行後保證與證據條件；
- P：權限與授權環境；
- S：前置世界狀態。

此判定比傳統：

> Γ|- y:T

多保存世界、權限、狀態、效果與後置證據。

本文區分六類效果：

- 值效果：產生新值；
- 狀態效果：讀寫權威狀態；
- 事件效果：建立或發布已發生事實；
- 外部效果：呼叫外部數位服務；

- 物理效果：作用於設備、資源與實體環境；
- 治理效果：授權、撤銷、批准、改規則與改責任。

本文指出，效果不應只被理解為「副作用」。副作用一詞常暗示效果是純計算以外的次要污染，但在真實程式中，付款、寄信、部署、授權、控制設備與記錄研究證據，往往正是程式目的本身。因此本文提出：

> 【Effect
> ≠
> Accidental Side Effect】

效果是程式如何與世界耦合的正式結構。

本文亦建立「效果保持」與「效果提升」。當高階語意被編譯到低階表示時，效果資訊不應消失。若某一高階節點標記為不可逆外部效果，任何低階生成碼、工作流與 Runtime 都應保留其授權、冪等、補償與證據要求。相反地，低階分析發現的外部寫入、網路呼叫與權限提升，也應回饋至高階語意與視圖。

本文使用條件判斷、資料庫更新、付款、網站發布、AI Agent 工具、醫療處置與物理控制等案例，說明相同語法在不同語意與效果環境中的差異。本文最後提出可證偽研究綱領，包括語法—語意錯配率、語意—效果漂移率、效果簽章完整度、隱藏效果事故率、權限與效果聯合檢查、效果保持、部分成功型別化、Runtime 觀測回饋及 AI 生成程式的三層驗證。

本文為下一篇〈意圖中介表示〉建立語言骨架：自然意圖必須先被轉換為可驗證語意結構，再產生明示效果與多重投影，而不能直接從一句話跳到不可逆世界行動。

關鍵詞： 語法、語意、效果系統、程式語意學、世界差分、權限、後置條件、效果保持

SECTION

Abstract

This paper develops a three-layer ontology of programming languages:

> mathcal L
> =
> <
> mathcal Y,
> mathcal M,
> mathcal E
> >

Syntax determines which structures can be formed. Semantics binds those structures to problem-world concepts, identities, states, and rules. Effects describe the actual digital, institutional, or physical world differences produced by execution.

The paper introduces three levels of validity, an extended typing and effect judgment, six classes of effects, effect preservation across compilation, and bidirectional feedback between high-level semantics and runtime-observed effects.

Keywords: syntax, semantics, effects, effect systems, world difference, programming language ontology

例如：

```
account.close()
```

這段文字可能：

- 語法合法；
- 名稱解析成功；
- 型別檢查通過；
- Runtime 可呼叫。

  但仍不知道：

- account 指向哪一個帳號；
- 關閉是暫停、停用還是永久刪除；
- 誰有權執行；
- 是否需通知；
- 是否保留歷史；
- 是否可恢復；
- 是否影響其他服務；
- 是否已產生法律與制度效果。

  因此，程式語言不能只在語法層理解程式。

本文定義：

> 【mathcal L
> =
> <
> mathcal Y,
> mathcal M,
> mathcal E
> >】

SECTION

2.1 語法層 mathcal Y

決定合法構造。

SECTION

2.2 語意層 mathcal M

決定構造在問題世界中的身份、關係與規則。

SECTION

2.3 效果層 mathcal E

決定執行後實際產生的狀態、事件、外部效果與證據。

SECTION

2.4 三層關係

> y
> xrightarrowllbracket·rrbracket
> m
> xrightarrowRun
> Δ W

文字或視覺結構先被解釋成語意，再由 Runtime 造成世界差分。

本文將語法層定義為：

> mathcal Y
> =
> <
> Σ,
> G,
> A,
> B
> >

SECTION

3.1 符號集合 Σ

包括：

- token；
- 關鍵字；
- 名稱；
- 常值；
- 操作符；
- 節點種類；
- 結構標籤。

SECTION

3.2 形成規則 G

決定：

- 哪些節點可包含哪些子節點；
- 哪些操作符具有何種參數；
- 哪些結構可嵌套；
- 哪些語法具有優先順序。

SECTION

3.3 抽象結構 A

語法經解析後形成：

- AST；
- 節點圖；
- 規則樹；
- 工作流；
- 狀態圖；
- 結構化表單。

SECTION

3.4 局部邊界 B

包括：

- 作用域；
- 名稱可見性；
- 模組；
- 區塊；
- 節點所有權；

- 局部參照。

SECTION

3.5 語法合法性

> Valid_(mathcal Y)(y)

只表示 y 屬於語言可形成結構。

它不保證：

- 名稱已正確綁定；
- 型別有效；
- 世界狀態允許；
- 權限存在；
- 效果安全；
- 目的合理。

SECTION

4.1 表面語法錯誤

例如：

- 括號未閉合；
- 關鍵字錯誤；
- 節點缺失；
- 非法標點；
- 不合法縮排。

SECTION

4.2 結構語法錯誤

即使字元可解析，結構也可能不完整：

- 行動缺少目標；
- 狀態轉移沒有來源；
- 事件沒有身份；
- 工作流沒有終止；
- 效果節點沒有證據出口。

SECTION

4.3 未完成結構

未完成程式可用 Hole 表示：

```
Box_T
```

Hole 可以攜帶：

- 預期型別；
- 世界位置；
- 需要的效果；
- 可用權限；
- 未決問題。

SECTION

4.4 語法應容納形成過程

成熟語言不應只接受完整與錯誤二分，而應支援：

draft
incomplete
unresolved
ambiguous
requires-binding
語意映射可表示為：

> llbracket y rrbracket_(Γ,W)
> =
> m

SECTION

5.1 環境 Γ

環境不只包含型別與變數，也包括：

- 身份；
- 作用域；
- 模組責任；
- 契約；
- 權限；
- 規則版本；
- 資料來源；
- 時間；
- Runtime 能力。

SECTION

5.2 問題世界 W

同一語法只有在某一問題世界中才具有具體意義。

例如 approve() 可能表示：

- 批准文件；
- 批准付款；
- 批准醫療處置；
- 批准部署；
- 批准 Agent 行動。

SECTION

5.3 綁定

語意綁定包括：

- 名稱到身份；
- 型別到世界類別；
- 操作到責任模組；
- 規則到權威來源；
- 狀態到合法生命週期；
- 效果到外部對象。

SECTION

5.4 語意不是註解

註解可補充說明，但若語意只存在於註解與作者腦中，編譯器、AI 與 Runtime 就無法驗證。

SECTION

6.1 指稱語意

描述結構指向什麼數學或計算對象。

> llbracket e rrbracket
> =
> v

SECTION

6.2 操作語意

描述程式如何一步步轉移：

> < e,S>
> →
> < e',S'>

SECTION

6.3 規範語意

描述：

- 誰可以執行；
- 哪些狀態合法；
- 哪些效果禁止；
- 哪些結果需要批准；
- 哪些主體有申訴權。

SECTION

6.4 世界語意

本文主張完整程式語意應同時包含：

> 【Meaning
> =
> Denotation
> +
> OperationalBehavior
> +
> NormativePosition
> +
> WorldBinding】

只描述值與步驟，仍不足以表示真實系統。

SECTION

7.1 同語法異語意

> llbracket y rrbracket_(Γ⊠,W⊠)
> ≠
> llbracket y rrbracket_(Γ⊠,W⊠)

例如：

user.delete()
在不同系統中可能表示：

- 刪除 UI 草稿；
- 軟刪除帳號；
- 停用登入；
- 匿名化個資；
- 永久清除身份。

SECTION

7.2 上下文必須可見

需要知道：

- 語言版本；
- 套件版本；
- 作用域；
- schema；
- 權限；
- 部署環境；
- 特性開關；
- 規則版本。

SECTION

7.3 隱含上下文風險

同一程式碼在測試與生產環境造成不同效果，通常不是語法差異，而是上下文與 Runtime 差異。

本文定義：

$$
\begin{aligned}
&\mathrm{Valid}_{\mathcal{M}}(m) \\
&\Leftrightarrow \\
&\mathrm{Bound}(m) \\
&\wedge \\
&\mathrm{Typed}(m) \\
&\wedge \\
&\mathrm{StateValid}(m) \\
&\wedge \\
&\mathrm{RuleConsistent}(m) \\
&\wedge
\end{aligned}
$$

SECTION

8.1 身份綁定

引用對象必須存在或被正式標示為未決。

SECTION

8.2 型別合法

不只資料型別，也包括：

- 單位；
- 幣別；
- 角色；
- 狀態；
- 事件；
- 證據類型。

SECTION

8.3 狀態合法

操作必須符合當前生命週期與版本。

SECTION

8.4 規則一致

不得在未解決衝突下同時滿足互斥規則。

SECTION

8.5 責任位置

必須知道哪個模組與主體對語意判定負責。

兩個語法結構 y☒,y☒ 可有：

> y☒
> ≠
> y☒

但：

> llbracket y☒ rrbracket_(Γ,W)
> ≡
> llbracket y☒ rrbracket_(Γ,W)

SECTION

9.1 重構

變數重命名、控制流改寫或抽象封裝，可能保持語意。

SECTION

9.2 語意等價的條件

需比較：

- 世界身份；
- 狀態前置；
- 規則；
- 輸入與輸出；
- 效果；
- 失敗；
- 權限；

- 證據。

SECTION

9.3 僅值等價不足

兩個程式回傳同一值，但一個寄信、一個不寄信，不能視為完整語意等價。

本文將效果定義為：

> 【mathcal E(m,R,S⊠)
> =
> <
> S_(t+1),
> Events,
> ExternalEffects,
> Evidence,
> Residual
> >】

SECTION

10.1 狀態差分

> Δ S
> =
> S_(t+1)
> -
> S⊠

包括權威狀態的建立、更新、終止、關係變化與版本變更。

SECTION

10.2 事件

執行可能形成：

- ActionAccepted；
- PaymentAuthorized；
- DeploymentStarted；
- PermissionRevoked；
- TreatmentAdministered。

SECTION

10.3 外部效果

包括：

- 網路呼叫；
- 寄送訊息；
- 金流；
- 發布；
- 寫入其他系統；
- 控制設備。

SECTION

10.4 證據

效果需具有：

- 回執；
- 事件身份；
- 狀態版本；

- 簽章；
- 感測確認；
- 人類接受。

SECTION

10.5 殘差

Residual 表示：

- 結果未知；
- 部分成功；
- 不可逆影響；
- 補償未完成；
- 未被模型完整表示的後果。

SECTION

11.1 副作用語言

在純函數中心語言中，狀態與 I/O 常被稱為 side effects。

SECTION

11.2 真實目的常是效果

對許多系統而言，真正目的就是：

- 完成付款；
- 更新權限；
- 發布網站；

- 寄送通知；
- 控制設備；
- 保存研究證據。

SECTION

11.3 正式命題

> 【Effect
> ≠
> Accidental Side Effect】

效果不是純計算旁邊的污染，而是程式與世界耦合的核心。

SECTION

11.4 純計算與世界耦合

純計算可被視為效果鏈中的可推理內部區域；效果邊界則負責真正世界差分。

SECTION

12.1 值效果

產生新值，但不修改權威世界。

SECTION

12.2 狀態效果

讀取、建立或修改權威狀態。

SECTION

12.3 事件效果

形成、發布或確認領域事件。

SECTION

12.4 外部數位效果

作用於：

- 第三方 API；
- 郵件；
- 金流；
- 雲端服務；
- 外部資料庫。

SECTION

12.5 物理效果

作用於：

- 馬達；
- 門鎖；
- 機器人；
- 感測器；
- 能源；
- 生產設備。

SECTION

12.6 治理效果

改變：

- 權限；
- 規則；
- 身份；
- 委任；
- 責任；
- 申訴位置；
- 可用行動空間。

本文提出：

> ε
> =
> <
> Read,
> Write,
> Emit,
> Call,
> Physical,
> Governance,
> Reversibility,
> Evidence
> >

SECTION

13.1 Read

讀取哪些資料與狀態，是否敏感。

SECTION

13.2 Write

修改哪些權威狀態。

SECTION

13.3 Emit

產生哪些事件。

SECTION

13.4 Call

呼叫哪些外部系統。

SECTION

13.5 Physical

是否可能造成物理效果。

SECTION

13.6 Governance

是否改變權限、規則與責任。

SECTION

13.7 Reversibility

效果是：

- 可精確回復；

- 數位可回滾；
- 可補償；
- 只能說明；
- 不可逆。

SECTION

13.8 Evidence

完成效果需提供什麼證據。

傳統型別判定：

> Γ
> |-
> y:T

本文擴展為：

> 【Γ;W;P;S
> |-
> y
> :
> T
> !
> ε
> triangleright
> Q】

SECTION

14.1 Γ

名稱、型別、身份與作用域環境。

SECTION

14.2 W

問題世界與規則版本。

SECTION

14.3 P

主體、權限、委任與批准環境。

SECTION

14.4 S

執行前世界狀態。

SECTION

14.5 T

值、事件、狀態或證據型別。

SECTION

14.6 ε

效果簽章。

SECTION

14.7 Q

執行後保證，包括狀態、事件、證據與殘差條件。

SECTION

15.1 語法合法

> Valid_(mathcal Y)(y)

SECTION

15.2 語意合法

> Valid_(mathcal M)
> (
> llbracket yrrbracket
>)

SECTION

15.3 效果合法

> Valid_(mathcal E)
> (
> ε,
> P,
> S,
> B
>)

其中 B 是系統責任與安全邊界。

SECTION

15.4 不可推出

> 【Valid_(mathcal Y)
> not⇒
> Valid_(mathcal M)
> not⇒
> Valid_(mathcal E)】

SECTION

15.5 執行門檻

只有三層均成立，且必要人類選擇完成後，程式才應取得實際執行權。

本文定義：

> Valid_(mathcal E)(ε)
> ⇔
> Authorized
> ∧
> Bounded
> ∧
> Observable
> ∧
> RecoverableOrAccepted
> ∧
> EvidenceDefined

SECTION

16.1 授權

主體是否可產生此效果。

SECTION

16.2 有界

影響是否位於明示作用域、資源與時間範圍。

SECTION

16.3 可觀測

是否能確認效果已發生或未發生。

SECTION

16.4 可恢復或已接受

可回復、可補償，或已經過必要主體明示接受不可逆性。

SECTION

16.5 證據

成功、失敗與未知必須具有可判定證據。

若：

> y□:T□!ε□

以及：

> y□:T□!ε□

其複合效果不是簡單集合相加。

SECTION

17.1 順序

> ε□ o ε□

可能不同於：

> ε☒ o ε☒

SECTION

17.2 效果依賴

第二效果可能依賴第一效果的證據或狀態。

SECTION

17.3 效果衝突

例如：

- 寫入後刪除；
- 授權後撤銷；
- 發布後回滾；
- 付款後取消。

SECTION

17.4 效果摘要

複合節點應推導：

- 整體讀寫集合；
- 外部服務；
- 不可逆節點；
- 權限聯集；
- 失敗聯集；

- 補償順序；
- 證據鏈。

SECTION

18.1 效果多態

高階函式可以接受不同效果算子，但仍保留效果參數：

> map
> :
> (A→ B!ε)
> →
> List(A)
> →
> List(B)!ε^ast

SECTION

18.2 效果限制

某環境可限定：

pure-only
read-only
no-external-call
no-governance-effect
human-approval-required

SECTION

18.3 沙盒

沙盒不是只隔離程序，也應限制效果能力。

SECTION

18.4 Agent 工具環境

Agent 的工具集合可視為其可用效果代數，但真正可執行集合仍受權限、預算與人類批准限制。

SECTION

19.1 名稱無法顯示效果

save()
process()
execute()
update()
可能隱藏完全不同世界影響。

SECTION

19.2 Getter 也可能有效果

一個看似讀取的方法可能：

- 觸發網路；
- 寫快取；
- 記錄追蹤；
- 更新最後存取時間；
- 產生費用。

SECTION

19.3 隱藏效果的後果

- 測試不穩定；
- 重試危險；
- 權限越界；
- 難以並行；

- 無法預覽世界差分；
- AI 誤用工具。

SECTION

19.4 效果透明原則

高風險外部、物理與治理效果必須能在結構、簽章或可查詢元資料中被識別。

SECTION

20.1 高階到低階

設高階結構 h 被降低為低階表示 l：

> Lower(h)=l

效果保持要求：

> Effects(l)
> succeq
> Effects(h)

也就是低階表示不能遺失高階已知的重要效果、權限與風險。

SECTION

20.2 不可逆效果保持

若高階節點標記：

irreversible
requires-approval
external-evidence-required

生成的程式碼、工作流與部署設定都必須保存這些要求。

SECTION

20.3 失敗保持

高階語意中的：

- 結果未知；
- 部分成功；
- 補償；
- 人工接管；

不能在低階投影中被壓成普通例外或布林值。

SECTION

20.4 證據保持

高階完成條件必須映射到實際可觀測證據。

SECTION

20.5 語意優化的限制

編譯器可重排、合併或消除純計算，但涉及外部效果時，需證明效果順序與觀測結果保持。

SECTION

21.1 靜態宣告可能不完整

第三方套件、動態載入與反射可能引入未宣告效果。

SECTION

21.2 Runtime 觀測

系統可觀測：

- 真實資料寫入；
- 網路呼叫；
- 權限使用；
- 事件發布；
- 物理命令；
- 延遲與失敗。

SECTION

21.3 效果提升

若 Runtime 發現：

$$
\begin{aligned}
&\text{ObservedEffects}(y) \\
&\supset \\
&\text{DeclaredEffects}(y)
\end{aligned}
$$

就應建立差異：

$$
\begin{aligned}
&\Delta\varepsilon \\
&= \\
&\text{ObservedEffects} \\
&- \\
&\text{DeclaredEffects}
\end{aligned}
$$

並回饋：

- 語意模型；
- 效果簽章；
- 安全規則；

- 測試；
- 視圖；
- AI 工具描述。

SECTION

21.4 宣告與觀測雙層

```
effects:
 declared:
  - "read-order"
 observed:
  - "read-order"
  - "write-access-log"
  - "call-external-analytics"
```

SECTION

21.5 未宣告效果

未宣告效果不必然惡意，但必須被審核，因為它可能改變隱私、費用、權限與恢復條件。

SECTION

22.1 單一回傳不足

外部工作流可能同時包含：

- 已完成；
- 失敗；
- 等待；
- 未知；
- 補償中。

SECTION

22.2 效果結果型別

可表示為：

> EffectResult
> =
> Success
> +
> Rejected
> +
> Partial
> +
> Unknown
> +
> Compensating
> +
> Irrecoverable

SECTION

22.3 Partial

```
partial:
 completed:
  - "payment-authorized"
 failed:
  - "inventory-reservation"
 unknown:
  - "notification-delivery"
```

SECTION

22.4 Unknown

結果未知時，語言應禁止未經查詢的重複不可逆效果。

SECTION

22.5 類型化恢復

不同結果型別只能接到相容恢復算子：

- RetrySafe；
- QueryOutcome；
- Compensate；
- Escalate；
- HumanReview。

SECTION

23.1 語法糖

語法糖改變表達便利，但理論上不改變核心語意。

SECTION

23.2 效果糖

某些簡寫隱藏複雜效果，例如：

```
publish(site)
```

可能展開為：

```
validate
build
upload
deploy
verify
shift-traffic
notify
```

SECTION

23.3 危險簡寫

如果簡寫隱藏：

- 不可逆效果；
- 多階段失敗；
- 人類批准；
- 外部依賴；
- 補償；

  就會降低世界可理解性。

SECTION

23.4 安全語法糖

安全簡寫應能展開並查看完整效果圖，而不是永久遮蔽。

語法：

```
if balance >= amount:
  approve()
```

SECTION

24.1 語法層

條件與呼叫合法。

SECTION

24.2 語意層

需知道：

- balance 是可用餘額還是帳面餘額；
- amount 的幣別；
- approve 批准哪項行動；
- 規則版本；
- 是否允許透支。

SECTION

24.3 效果層

approve() 可能：

- 只回傳值；
- 寫入授權狀態；
- 呼叫銀行；
- 形成法律承諾。

同一語法外觀可能跨越完全不同效果層級。

```
UPDATE users SET active = false WHERE id = ?;
```

SECTION

25.1 語法

SQL 合法。

SECTION

25.2 語意

active=false 可能表示：

- 禁止登入；

- 暫停服務；
- 刪除帳號；
- 法規封鎖；
- 使用者自行退出。

SECTION

25.3 效果

還可能需要：

- 撤銷 token；
- 停止排程；
- 保留資料；
- 通知使用者；
- 發布 AccountSuspended；
- 允許申訴。

單一欄位更新不能自動代表完整世界效果。

語法：

```
await charge(card, amount)
```

SECTION

26.1 語意

需要綁定：

- 客戶；
- 訂單；
- 幣別；

- 金額權威來源；
- 付款嘗試身份；
- 冪等鍵。

SECTION

26.2 效果

包括：

- 外部金流；
- 費用；
- 付款事件；
- 結果未知；
- 可能補償；
- 財務證據。

SECTION

26.3 合法性

即使函式型別正確，若沒有授權、冪等與結果查詢，仍不可安全執行。

語法：

publish website

SECTION

27.1 語意展開

- 哪個版本；
- 哪個環境；

- 哪個網域；
- 哪些批准；
- 哪些完成條件。

SECTION

27.2 效果鏈

build
→ upload
→ deploy
→ verify
→ shift traffic
→ expose public content

SECTION

27.3 不可逆部分

公開內容可能已被：

- 爬蟲抓取；
- 使用者閱讀；
- 外部快取；
- 搜尋引擎索引。

回滾部署不等於撤回所有世界效果。

工具宣告：

delete_file(path)

SECTION

28.1 語法與型別

只需要一個路徑字串。

SECTION

28.2 語意

需知道：

- 路徑屬於哪個工作區；
- 檔案身份；
- 是否為生成物；
- 是否有版本或備份；
- 是否為他人所有。

SECTION

28.3 效果

- 刪除；
- 可能不可恢復；
- 破壞相依；
- 改變發布；
- 影響其他 Agent。

SECTION

28.4 Agent 執行門檻

需要效果預覽、權限、備份與必要的人類批准，而非只通過參數型別。

語法節點：

administer-treatment

SECTION

29.1 語意

必須綁定：

- 病人；
- 治療；
- 醫師；
- 同意；
- 劑量；
- 時間；
- 適用條件。

SECTION

29.2 效果

物理效果可能不可逆，且不能只依系統內部成功確認。

SECTION

29.3 證據

需要：

- 專業批准；
- 病人同意；
- 執行紀錄；
- 真實處置確認；

- 後續觀測。

door.unlock()

SECTION

30.1 語法

方法呼叫合法。

SECTION

30.2 語意

需知道：

- 哪一扇門；
- 哪個區域；
- 誰要求；
- 現在是否緊急；
- 是否有在場人員。

SECTION

30.3 效果

命令送出不代表門鎖已打開。

需要感測確認與失敗狀態：

unlock-command-sent
unlock-confirmed
unlock-outcome-unknown
mechanical-failure

SECTION

30.4 治理效果

開門也改變了物理空間的可進入權限。

- 語法正確即程式正確： 可解析被誤認為完整合法。
- 型別正確即世界合法： 忽略狀態、權限與規則。
- 名稱即語意： 不做世界身份與責任綁定。
- 語意只存在註解： 編譯器、AI 與 Runtime 無法驗證。
- 上下文隱形： 版本、環境、特性開關與規則來源缺席。
- 值等價即程式等價： 忽略外部效果與證據。
- 效果等於副作用： 將程式真正目的降為次要污染。
- 普通函式隱藏高風險效果： 付款、發布與治理無法被辨識。
- Getter 純度假設： 讀取操作暗中產生寫入、費用與追蹤。
- 授權在語言外部： 結構取得執行權前無法聯合檢查。
- 部分成功布林化： 多階段世界狀態被壓成 true／false。
- 結果未知錯誤化： 超時後直接重試不可逆效果。
- 效果順序無語意： 編譯器或開發者錯誤重排外部操作。
- 效果集合簡單相加： 忽略順序、衝突與相依。
- 低階投影遺失效果： 高階不可逆與批准要求在生成碼中消失。
- Runtime 未宣告效果不回饋： 實際行為長期偏離語意模型。
- 語法糖隱藏責任： 一行簡寫遮蔽多階段失敗與補償。
- 沙盒只隔離程序： 未限制資料、外部與治理效果。
- AI 只做語法驗證： 生成碼可編譯卻越權或不可恢復。
- 物理效果無感測證據： 命令送出被當成世界完成。
- 治理效果不可見： 權限與規則修改無法進入效果分析。

- 語意與效果版本分離： 語意宣告未隨 Runtime 行為更新。

SECTION

32.1 語法—語意錯配率

統計可解析、可型別檢查的結構中，有多少仍指向錯誤身份、狀態、規則或責任。

SECTION

32.2 語意—效果漂移率

比較宣告語意與 Runtime 真實效果：

> R_(ME)
> =
> (|observed effects absent from semantic declaration|)/(|observed effects|)

SECTION

32.3 效果簽章完整度

對效果八元簽章評分，研究完整度與事故、重試錯誤、越權及恢復時間的關係。

SECTION

32.4 隱藏效果事故率

統計看似純粹或低風險操作中，由未宣告網路、寫入、費用、發布或治理效果造成的事故。

SECTION

32.5 權限—效果聯合檢查

比較只做權限中介層檢查，以及把權限與效果簽章納入語言判定的系統。

SECTION

32.6 部分成功型別化

測量使用正式 Partial／Unknown／Compensating 型別，是否降低盲目重試與錯誤完成敘事。

SECTION

32.7 效果保持

對高階語意到低階程式碼、工作流與部署投影，檢查不可逆性、批准、失敗與證據是否保持。

SECTION

32.8 效果重排安全性

比較只依資料依賴重排，以及同時考慮效果順序、外部觀測與治理條件的最佳化。

SECTION

32.9 Runtime 回饋閉環

測量觀測到未宣告效果後，能否自動建立差異、阻止執行、更新模型或觸發審核。

SECTION

32.10 語意差分預測

比較文字 diff、AST diff、語意 diff 與效果 diff 對實際世界影響的預測能力。

SECTION

32.11 AI 三層驗證

比較 AI 直接生成可編譯程式，以及依序通過語法、語意、效果驗證後再執行的錯誤率。

SECTION

32.12 沙盒效果約束

測量只隔離 CPU／檔案程序，與同時限制網路、資料、費用、權限及治理效果的沙盒差異。

SECTION

32.13 外部證據完整率

統計系統宣稱完成的外部與物理效果中，有多少具有足夠回執、感測或主體接受證據。

SECTION

32.14 三層教學實驗

比較只教語法與型別，及同時教授問題世界語意和效果簽章的學習者，在真實系統設計上的表現。

- 程式語言具有語法、語意與效果三層存在。
- 語法回答哪些結構可以形成，不回答世界是否合理。
- 語意將結構綁定到問題世界身份、狀態、規則與責任。
- 效果描述 Runtime 對數位、制度與物理世界造成的差分。

5.

> Syntax
> ≠
> Semantics
> ≠
> Effects

- 語法合法不推出語意合法，語意合法不推出效果合法。
- 同一語法在不同上下文與世界中可具有不同語意。

- 語意不只包含值與操作步驟，也包含規範位置與世界綁定。
- 兩個程式回傳同值，不代表具有相同語意與效果。
- 效果不是純計算之外的偶然副作用，而是程式與世界耦合的正式結構。
- 效果至少應區分值、狀態、事件、外部、物理與治理六類。
- 完整語言判定應包含世界、狀態、權限、效果與後置證據。
- 效果合法性要求授權、有界、可觀測、可恢復或已接受，以及證據明確。
- 複合效果不能只以集合聯集描述，還需保存順序、依賴、衝突與補償。
- 高階函式與 Agent 工具應保留效果多態與效果限制。
- 部分成功、結果未知與補償中應成為正式效果結果型別。
- 高階語意降低為低階表示時，不得遺失不可逆性、批准、失敗與證據。
- Runtime 發現未宣告效果時，應回饋語意模型與治理。
- 宣告效果與觀測效果的差異，是安全、隱私與維護的重要訊號。
- 語法糖可以簡化表達，但不能永久遮蔽高風險世界責任。
- 物理命令與治理操作必須具有比一般值運算更嚴格的效果判定。
- AI 生成程式不能只通過語法與型別檢查。
- 真正的程式審查應同時審查結構、意義與世界差分。

24.

> 【程式語言不是只告訴機器如何形成指令，】
>
> 【而是必須同時說明這些指令代表什麼，】
>
> 【以及它們取得執行權後，】

【世界將因此發生什麼。】

SECTION

34.1 承接 PU-2-01

PU-2-01 將文字定位為語言的一種投影。

本篇進一步指出，無論文字或視覺投影，都必須通過語法、語意與效果三層。

SECTION

34.2 承接 PU-2-02

PU-2-02 將符號定義為受治理算子。

本篇分解算子的三個核心面向：

- 語法：算子如何形成與組合；
- 語意：算子指向何種世界變換；
- 效果：算子真正造成何種世界差分。

SECTION

34.3 承接第 1 冊

第 1 冊建立的狀態、事件、責任、契約、失敗與恢復，現在可以正式進入語意環境與效果簽章。

SECTION

34.4 銜接 PU-2-04

下一篇將建立意圖中介表示：

Intent
→
SemanticIR

集中處理：

- 自然意圖如何被分解；
- 歧義如何被保存與消除；
- 語意節點如何綁定問題世界；
- 效果、權限與證據如何先於生成碼；
- 同一 IR 如何生成程式碼、UI、工作流、測試與文件。

傳統語言工具首先判斷程式能否被解析。

型別系統進一步判斷值與結構能否正確組合。

這些能力極為重要，但對真實世界程式仍不完整。

因為一段程式即使：

- 可解析；
- 型別正確；
- 測試通過；
- Runtime 可執行；

仍可能：

- 指向錯誤身份；
- 使用過期狀態；
- 違反責任邊界；
- 未經授權；
- 造成不可逆效果；
- 缺少外部證據；
- 在部分成功後錯誤重試；

- 把系統成功冒充目的完成。

  所以程式語言的合法性不能停在語法層。

  本文將完整語言判定收束為：

> 【Program Validity
> =
> Syntactic Validity
> +
> Semantic Validity
> +
> Effect Legitimacy】

而程式執行則是：

> 【Execution
> =
> Meaning Bound to a World
> +
> Authority to Act
> +
> Observable World Difference】

語法提供可構造性。

語意提供可理解性。

效果提供世界耦合。

三者缺一，程式語言都無法成為完整的世界建構語言。

如果只有語法，語言只是合法符號機器。

如果只有語意，語言只是尚未執行的世界描述。

如果只有效果而缺少語意與責任，系統就可能成為無法解釋與治理的世界修改器。

因此，第 2 冊至此形成新的中心結構：

> 【Symbol
> →
> Syntax
> →
> Semantics
> →
> Effects
> →
> World Difference】

本文的最終命題是：

> 【好的程式語言，】

> 【不只拒絕無法形成的句子，】

> 【也應拒絕無法理解、無權執行、】

> 【無法驗證或無法承擔的世界效果。】

```
language_node:
 node_id: "publish-site"
 syntax:
  node_type: "effect-action"
  required_children:
   - "site"
   - "release"
   - "environment"
 semantics:
  world_role: "make approved release publicly accessible"
  responsibility_module: "deployment"
  preconditions:
   - "release is approved"
   - "environment is production"
 effects:
  writes:
   - "deployment-state"
  external:
```

```
    - "upload artifact"
    - "shift public traffic"
  irreversible:
    - "content may be externally cached"
  evidence:
    - "deployment receipt"
    - "public journey verification"
language_judgment:
 expression: "AuthorizePayment(order-932)"
 environment:
  world: "commerce-v4"
  state_version: 18
  actor: "order-module"
  permissions:
   - "request-payment-authorization"
 type:
  output: "PaymentAuthorizationResult"
 effects:
  reads:
   - "authoritative-order-total"
  external_calls:
   - "payment-provider"
  events:
   - "PaymentAuthorized"
   - "PaymentDeclined"
   - "PaymentOutcomeUnknown"
  reversibility: "compensatable"
 postconditions:
  evidence_required:
   - "provider-receipt"
effect_diff:
 node: "load-user-profile"
 declared:
  - "read user profile"
 observed:
  - "read user profile"
  - "write last-accessed-at"
  - "call analytics provider"
 undeclared:
  - "write last-accessed-at"
  - "call analytics provider"
 governance:
  review_required: true
  privacy_impact: true
effect_result:
```

```
kind: "partial"
completed:
  - effect: "release-deployed"
    evidence: "deployment-889"
failed:
  - effect: "mobile-smoke-test"
    reason: "authentication failure"
unknown:
  - effect: "dns-propagation"
allowed_next:
  - "hold-traffic"
  - "rollback-deployment"
  - "human-review"
forbidden_next:
  - "declare-publication-complete"
```

- PU-2-01　程式語言不等於文字：結構、語意與執行的基本分離
- PU-2-02　符號作為算子：從靜態字元到可組合計算閉包
- PU-2-03　語法—語意—效果：程式語言的三層存在結構
- PU-2-04　意圖中介表示：從自然意圖到多重可執行投影
- PU-2-05　可編譯世界：程式執行作為世界狀態差分
- PU-2-06　後文本程式語言：意圖、結構、驗證與物理耦合的統一框架

SECTION

Neo.K／EveMissLab 相關理論

- Neo.K with Aletheia，《程式語言不等於文字：結構、語意與執行的基本分離》，2026。
- Neo.K with Aletheia，《符號作為算子：從靜態字元到可組合計算閉包》，2026。
- Neo.K with Aletheia，《程式不等於程式碼：可執行問題模型的基本定義》，2026。
- Neo.K with Aletheia，《資料—狀態—事件—行動：程式系統的四元動力結構》，2026。
- Neo.K with Aletheia，《責任、模組與契約：從檔案分類到系統邊界》，2026。
- Neo.K with Aletheia，《失敗也是程式：驗證、可觀測、恢復與長期維護》，2026。

- Neo.K with Aletheia，《可編譯世界：從程式執行到世界狀態演化》，2026。

SECTION

一般理論背景

- Scott, D. and Strachey, C., “Toward a Mathematical Semantics for Computer Languages,” 1971.
- Plotkin, G. D., “A Structural Approach to Operational Semantics,” 1981.
- Hoare, C. A. R., “An Axiomatic Basis for Computer Programming,” 1969.
- Milner, R., “A Theory of Type Polymorphism in Programming,” 1978.
- Pierce, B. C., Types and Programming Languages, 2002.
- Reynolds, J. C., Theories of Programming Languages, 1998.
- Moggi, E., “Notions of Computation and Monads,” 1991.
- Wadler, P., “The Essence of Functional Programming,” 1992.
- Plotkin, G. D. and Power, J., “Algebraic Operations and Generic Effects,” 2003.
- Koka language documentation and research on row-polymorphic effect types.
- Bauer, A. and Pretnar, M., “Programming with Algebraic Effects and Handlers,” 2015.
- Nielson, F., Nielson, H. R., and Hankin, C., Principles of Program Analysis, 1999.
- Winskel, G., The Formal Semantics of Programming Languages, 1993.

SECTION

v0.1 — 2026-07-27

- 完成十八篇地基論文第十五篇。
- 建立語法、語意與效果三層存在模型。

- 將語法定義為符號、形成規則、抽象結構與局部邊界。
- 建立問題世界與上下文條件下的語意映射。
- 擴展指稱、操作、規範與世界語意。
- 建立語意合法性與語意等價條件。
- 將效果定義為狀態、事件、外部效果、證據與殘差。
- 建立值、狀態、事件、外部、物理與治理六類效果。
- 建立效果八元簽章。
- 將傳統型別判定擴展為世界、權限、狀態、效果與後置證據判定。
- 建立三層合法性與效果合法性條件。
- 分析效果組合、多態、限制與隱藏效果。
- 建立效果保持、效果提升與 Runtime 回饋。
- 將部分成功與結果未知納入效果型別。
- 加入條件、資料庫、付款、發布、AI Agent、醫療與物理控制案例。
- 提出二十二類失敗模式與十四項可證偽研究方向。
- 完成與 PU-2-04《意圖中介表示》的銜接。