5. Confluence

합류성(Confluence):

동일한 항 t가 여러 경로로 축약되더라도 **결국 같은 결과(항 u)**에 도달함을 보장하는 성질.

→ 즉, **평가 결과가 일관됨**을 의미함.

Example 5.1

식: $\{f(f(x)) = g(x),\ a = b\}$
항 $f(f(f(a)))$에서 첫 번째 방정식은

- "위치 ε (맨 위)"
- "위치 **1** (내부)
  모두에서 적용 가능."

→ 어떤 위치에서 적용하든 결과가 같다면 **합류성**이 있음.

Definition 5.1 — Confluence

명세 (Σ,E)는 다음 조건일 때 **합류적(confluent)** 이다.

$$\forall t, t_1, t_2 \text{ such that } t \xrightarrow{*} t_1,\ t \xrightarrow{*} t_2,\ \exists u : t_1 \xrightarrow{*} u,\ t_2 \xrightarrow{*} u$$

- **Ground confluence**: 위 성질이 모든 ground term에 대해 성립할 때.

Example 5.2

- 예제 5.1의 명세는 **합류적이지 않음**

$$f(f(f(x))) \to f(g(x)), \quad f(f(f(x))) \to g(f(x))$$

  → f(g(x))와 g(f(x))는 더 이상 줄일 수 없고, 공통 항이 없음.

- 해결법:

  방정식 f(g(x))=g(f(x))를 추가하면 종료적이며 합류적인 시스템이 됨.

실제 검증의 어려움

합류성 직접 검사(모든 t, t_1, t_2는 불가능함:

1. 가능한 t_1, t_2가 너무 많음.
2. 가능한 t가 무한히 많음.

축소 필요

(i) 제한된 수의 t_1, t_2만 고려

(ii) 유한 개의 시작 항 t만 고려

Definition 5.2 — Local Confluence (국소 합류성)

명세가 **국소 합류적(local confluent)** 이란,

$$t \to t_1,\ t \to t_2 \implies \exists u : t_1 \xrightarrow{*} u,\ t_2 \xrightarrow{*} u$$

즉, **한 단계(one-step)** 축약만으로 검사 가능.

<그림 5.1 오른쪽> 참고.

Theorem 5.1 — Newman's Lemma

종료적(terminating)이고 국소 합류적(local confluent)인 명세는 합류적(confluent) 이다.

5.1 Unification

Definition 5.3 — Unifier

두 항 t, u의 **통일자(unifier)**는 다음 조건을 만족하는 치환(substitution) σ : tσ = uσ

예제 (Example 5.3)

✅ 통일 가능한 경우

$$f(x, h(b)) \text{ 와 } f(h(y), z)$$

→ 통일자:

$$\sigma = \{x \mapsto h(y),\ z \mapsto h(b)\}$$

이외의 인스턴스들도 통일자임:

$$\sigma' = \{x \mapsto h(f(f(a,a),a)),\ y \mapsto f(f(a,a),a),\ z \mapsto h(b)\}$$

🚫 통일 불가능한 경우

$$f(g(x)) \text{ 와 } f(h(z))$$

→ 함수기호 불일치 $g \neq h \Rightarrow$ unifiable 아님

가장 일반적인 통일자 (Most General Unifier, MGU)

- 여러 통일자 중에서 **가장 일반적인(universal)** 형태
- 다른 모든 통일자 σ는 ρ의 인스턴스로 표현됨:

  σ = π ∘ ρ

  (∘: 함수 합성, (f ∘ g)(x) = f(g(x)))
- 즉, **MGU**는 "모든 통일자를 생성할 수 있는 최소한의 통일자"

예제 (Example 5.3 cont.)

$$\sigma = \{x \mapsto h(y),\ z \mapsto h(b)\}$$

은 $f(x, h(b))$ 와 $f(h(y), z)$의 MGU.

다른 통일자:

$$\sigma' = \{y \mapsto f(f(a,a),a)\} \circ \sigma$$

$$\sigma'' = \{y \mapsto h(h(h(h(z))))\} \circ \sigma$$

정리 (Proposition 5.1)

두 항이 통일 가능하다면, 항상 MGU가 존재한다.

또한, MGU는 **변수 이름 변경(rename)**에 대해 **유일(unique)**하다.

변수 이름 바꾸기 (Renaming)

변수 이름만 바꾼 식은 논리적으로 동일함:

$$f(x,y),\ f(x',y'),\ f(y,x)$$

모두 "같은 식(renamed)"
→ 명세의 논리를 바꾸지 않음.

예시:

$$\{f(x,y) = g(x),\ h(x,y) = f(x,y)\} \equiv \{f(x,z) = g(x),\ h(x',y') = f(x',y')\}$$

Martelli–Montanari Unification Algorithm

두 항이 unifiable한지 확인하고, MGU를 계산하는 알고리즘.

알고리즘 초기 상태

(UP, ρ)

- UP: 통일 문제들의 집합 (t ≐ u)
- ρ: 현재까지 구성된 통일자 (초기값 = 항등치환 (Id))

알고리즘 절차

단계	조건	동작
1	f(t_1,...,t_n) ≐ g(u_1,...,u_m), f ≠ g	**Not unifiable**
2	f(t_1,...,t_n) ≐ f(u_1,...,u_n)	인자 분해: UP := {t_i ≐ u_i} ∪ UP'
3	t ≐ t	제거
4	x ≐ t이고 x가 t 안에 존재	**Not unifiable (순환)**
5	x ≐ t, x ∉ t	x ↦ t로 모든 항에 대체 및 ρ 갱신
6	UP = ∅	반환: ρ = MGU

예제 5.4

구하려는 MGU:

$$f(x, h(x)) \text{ 와 } f(h(y), z)$$

초기 상태:

$$(\{f(x, h(x)) \stackrel{?}{=} f(h(y), z)\},\ Id)$$

$$Id = \{x \mapsto x, y \mapsto y, z \mapsto z\}$$

단계별 진행

단계	상태	설명
(1)	$\{f(x, h(x)) \stackrel{?}{=} f(h(y), z)\}$	Step 2 — 인자 분해
(2)	$\{x \stackrel{?}{=} h(y),\ h(x) \stackrel{?}{=} z\}$	Step 5 — $x \mapsto h(y)$ 대입
(3)	$\{h(h(y)) \stackrel{?}{=} z\},\ \{x \mapsto h(y)\}$	Step 5 반복 적용
(4)	$\emptyset,\ \{x \mapsto h(y),\ z \mapsto h(h(y))\}$	Step 6 종료

✅ 결과:

$$\boxed{\rho = \{x \mapsto h(y),\ z \mapsto h(h(y))\}}$$

정리

- 알고리즘은 **정확성(correctness)** 과 **종료성(termination)** 이 입증됨. (참조: [105])

5.2 Checking Local Confluence

핵심 개념 요약

- **합류성(Confluence)**:

  항 `t` 가 여러 방식으로 줄어들더라도, 결국 같은 항 `u` 로 합쳐질 수 있는 성질.

- **Newman's Lemma**:

  시스템이 **종료적(terminating)**이면, 전체 합류성을 보기 위해

  한 단계(one-step)만의 분기(local confluence)만 검사하면 충분하다.

Local Confluence 검사 방법

1. **두 개의 방정식 선택**
   - 서로 다른(또는 같은) 식 두 개를 선택.
   - 변수 이름이 겹치지 않게 변경 (예: `x` → `x'`).
2. **왼쪽항의 겹침(overlap) 확인**
   - 한 식의 왼쪽항 전체(`l_j`)가 다른 식의 왼쪽항(`l_i`)의 **비변수 부분항(non-variable subterm)**과 통일(unify)될 수 있는지 확인.

- 통일되면 그 지점을 “overlap term”이라 함.

3. **두 가지 축약(reduction) 수행**
   - (1) `l_ip → r_ip` (맨 위에서 식 i 적용)
   - (2) `l_ip → (l_ip)[r_jp]_p` (내부 위치 p에서 식 j 적용)

   ⇒ 이렇게 얻은 두 항이 **joinable(합류 가능)** 한지 확인.
4. **모든 식 쌍과 모든 겹치는 위치에 대해 반복**
   - 모든 경우에서 joinable하면 명세는 **locally confluent**.
   - joinable하지 않은 경우가 하나라도 있으면 **not confluent**.

Theorem 5.2 — Critical Pair Lemma

A specification is locally confluent if and only if all its critical pairs are joinable.

즉, **모든 임계쌍(critical pair)** `(r_ip, (l_ip)[r_jp]_p)` 이 합류 가능하면 시스템 전체가 **locally confluent** 하다

Example 5.5 — 비합류적(non-confluent) 사례

명세: f(f(x)) = g(x)

과정

1. 식을 복제하고 변수 이름만 다르게:

   f(f(x)) = g(x)

   f(f(x')) = g(x')
2. 왼쪽항 `f(f(x))` 의 부분항 `f(x)` 와 `f(f(x'))` 을 통일

   → MGU: `ρ = {x ↦ f(x')}`

   → 겹치는 항(overlap term): `f(f(f(x')))`
3. 두 가지 축약 발생:
   - top-level: `f(f(f(x'))) → g(f(x'))`
   - inner: `f(f(f(x'))) → f(g(x'))`
4. critical pair: `(g(f(x')), f(g(x')))`
   - 두 항 모두 더 이상 줄어들 수 없음
   - **joinable하지 않음**

- 결론:

  f(f(x)) = g(x)는 비합류적 (not confluent)

Example 5.6 — 합류적(confluent) 사례

명세: 0 + x = x, s(x) + y = s(x + y)

- 각 식의 왼쪽항 사이에 **비자명한 겹침(non-trivial overlap)**이 없음
- 따라서 **합류적(confluent)**.

Equational Completion (방정식 완비화 과정)

Equational Completion: 비합류적 명세를 "논리적으로 동등하면서" 합류적이고 종료적인(confluent + terminating) 명세로 바꾸는 절차.

방법 : joinable하지 않은 critical pair `(t, u)` 가 생기면 새로운 방정식 `t = u` 를 명세에 추가.

예시 :

- Example 5.5의 비합류적 상황 : (g(f(x')), f(g(x')))
- 새로운 식 추가 : f(g(x)) = g(f(x))
- 시스템이 **confluent + terminating** 하게 됨.
- 단, 새로운 식을 추가한 후에는 다시 한 번 **합류성과 종료성**을 검사해야 함. (새로운 critical pair가 생길 수 있기 때문)