

7. Models of Equational Specifications

핵심 개념

- 목표 :

Maude 명세가 정의하는 **수학적 모델**(mathematical model)을 이해하고, 이를 **대수적 구조**(algebraic structure)로 정식화한다.

- 핵심 개념 :

프로그램(또는 명세)이 정의하는 수학적 객체 = **그 프로그램의 의미론**(denotational semantics)

→ Maude의 수학적 의미를 대수로 정의하는 것이 이 장의 목적.

Equational Model의 개념

- 등식 명세 (Σ, E) 는 여러 **모델**(models)을 가질 수 있다.
- 그러나 대부분의 경우 관심 있는 것은 **의도된 모델**(intended model) 하나이다.
- 이 모델은 우리가 명세를 통해 의도한 “진짜 수학적 구조”를 의미한다.

Equational Logic의 성질

- 등식 논리는 **sound**(논리적으로 참인 것은 모델에서도 참)하고 **complete**(모델에서 참인 것은 논리적으로 증명 가능)함. 즉,

$$t_1 = t_2 \text{ holds in all models of } (\Sigma, E) \iff E \vdash t_1 = t_2$$

- E가 **수렴**(confluent) 하고 **종결**(terminating) 하면, $E \vdash t_1 = t_2$ 판정은 **결정 가능**(decidable) 하다.
- 하지만 “의도된 모델” 내에서 등식이 성립하는지를 판정하는 문제는 **귀납적 정리**(inductive theorem) 문제로, 일반적으로 **완전한 증명 체계가 없음**.

Algebra로서의 소프트웨어 모듈

- Meseguer & Goguen: 소프트웨어 모듈은 여러 데이터 집합(sets)과 그 위의 연산(operations)을 포함하므로, 본질적으로 대수(algebra)의 구조를 가진다.
- 따라서
 - **Maude 명세 = 대수(algebra)를 정의하는 것**
 - Maude의 함수형 모듈은 결국 수학적 대수를 명세한다.

Intended Model (초기 대수, Initial Algebra)

- 7.4절:
 - 의도된 모델은 (Σ, E) -모델들의 집합 $\text{Alg}(\Sigma, E)$ 안의 **초기 대수**(initial algebra)로 정의된다.
- 이 모델은 유일하지 않지만, 같은 구조를 가지며 단지 **표현 방식만 다름**.
 - 예: 불리언의 경우
`true/false`, `T/F`, `1/0` 모두 동일한 구조를 표현.
- 따라서 (Σ, E) 는 **추상 자료형**(ADT, abstract data type)을 정의하며, 구체적인 표현(concrete representation)은 정의하지 않는다.

초기 대수의 성질

- 모든 명세는 의도된 모델을 가진다.
- 이 모델들은 서로 **동형(isomorphic)**인 구조를 가진다.
- (Σ, E) 는 초기 모델 $T_{\Sigma, E}$ 을 가지며, 이는 **ground term들의 동치류**로 표현된다.
- 올바른 의도된 모델은 다음 조건을 만족해야 한다:
 - "쓸모없는(junk)" 원소 없음 (모든 원소는 어떤 ground term으로 표현됨)
 - E-동치인 ground term들은 같은 값을 가짐

단순화된 가정

- 설명의 단순화를 위해,
 - 단일 정렬(un-sorted)
 - 조건부 등식(conditional equations)과 함수 속성 없음
- 다중 정렬(many-sorted) 확장은 ground term이 있으면 간단히 가능.

섹션별 개요

절	내용 요약
7.1	Σ -대수와 Σ -homomorphism 정의
7.2	$\text{Alg}(\Sigma, E)$: (Σ, E) 를 만족하는 모든 대수들의 집합
7.3	등식 논리의 건전성(soundness)과 완전성(completeness) 증명
7.4	의도된 모델(초기 대수)의 정의 및 성질
7.5	다중 정렬 등식 논리에 대한 간단한 확장 논의

7.1 Many-Sorted Σ -Algebras

Maude 명세와 대수(Algebra)의 관계

- **Maude 코드** (fmod 등)는 실제 계산을 정의하지 않고, "이런 연산들이 이런 등식을 만족해야 한다"라는 **명세(specification)**만 제공.
- 이 명세를 만족하는 **수학적 구조(대수, algebra)**들이 Maude 명세의 **모델(models)**이다.
- 즉, Maude 명세 $\rightarrow (\Sigma, E) \rightarrow$ 그걸 만족하는 대수들(모델들)

정의

Many-sorted signature $(\Sigma = (S, \Sigma))$ 에 대한 Σ -Algebra A 는 다음으로 구성됨:

1. 각 정렬 $s \in S$ 마다 하나의 집합 A_s
2. 각 함수기호 $f \in \Sigma_{s_1, \dots, s_n, s}$ 마다 실제 함수

$$f_A : A_{s_1} \times \dots \times A_{s_n} \rightarrow A_s$$

즉, 정렬은 "값의 집합", 연산은 "그 집합 위의 실제 함수".

Example 7.1 — NAT< 모듈의 해석 (표준 모델)

요소	해석
$A_{\text{Nat}} = \mathbb{N}, A_{\text{Bool}} = \{t, f\}$	실제 값 집합
$0_A = 0$	상수 0
$s_A(n) = n + 1$	successor 함수
$+_A(m, n) = m + n$	덧셈 함수
$<_A(m, n) = t \text{ if } m < n \text{ else } f$	비교 연산
$\text{true}_A = t, \text{false}_A = f$	불리언 상수

→ 이것이 **표준(의도된) 모델**, 즉 **초기 대수(initial algebra)**

Maude가 실제로 계산할 때 쓰는 수학적 의미다.

Example 7.2 — 비표준 모델(Non-standard Model)

Maude의 **NAT<** 서명을 문자열 기반으로 **다르게 해석**할 수도 있다.

구성	해석
$S_{\text{Nat}} = \{a, b, c\}^*$	문자열들의 집합
$S_{\text{Bool}} = \{t, f\}$	불리언
$0_S = \varepsilon$	빈 문자열
$s_S(x) = x + "a"$	문자열 뒤에 "a" 추가
$+_S$	문자열 이어붙이기
$<_S$	부분문자열(substring) 관계

→ **NAT<** 명세를 만족하지만, 의미는 전혀 다름.

즉, **같은 Maude 코드라도 무한히 많은 모델이 존재**할 수 있다.

Example 7.3 — NAT-ADD 명세를 만족하는 여러 대수들

이 부분이 핵심.

"같은 명세(NAT-ADD)"를 만족하는 수많은 대수 모델을 예시로 보여줌.

번호	대수 이름
(1) \mathbb{N}	표준 자연수 모델 (0, +, s)
(2) N_{\perp}	정의되지 않은 값 \perp 추가
(3) $N_{\times N}$	쌍(pair)을 이용한 비표준 해석
(4) \mathcal{E}	짝수만 포함하는 모델
(5) \mathbb{Z}	정수 전체
(6) squares	제곱수만 포함, 특수 연산 정의
(7) bits	비트열 기반 덧셈
(8) bits0	leading zero 허용 비트열 모델
(9) *	단 하나의 원소만 가진 대수
(10) +2	successor가 "+2"로 정의된 모델
(11) $\mathbb{Q}_{\geq 0}$	비음이 아닌 유리수 덧셈군
(12) N_k	mod k 연산 기반 모델
(13) AB	{*, a, b} 세 원소를 가진 특이한 대수

→ 결론 : 하나의 Maude 명세는 **무수히 많은 수학적 모델**을 가질 수 있고, 그 중 "초기 대수(initial algebra)"가 **Maude가 실제로 사용하는 의미론**이다.

Example 7.4 — Groups의 서명과 여러 대수들

groups 명세:

- sort: s
- 상수 e (항등원)
- 단항 함수 i (역원)
- 이항 함수 \circ (군 연산)

대수	도메인	해석
(1) \mathbb{Z}	정수	$e = 0, i(x) = -x, x \circ y = x + y$
(2) $\mathbb{R}_{>0}$	양의 유리수	$e = 1, i(x) = 1/x, x \circ y = x * y$
(3) $\text{funcs}(\{a,b,c\})$	$\{a,b,c\} \rightarrow \{a,b,c\}$ 전단사 함수 집합	$e = \lambda x.x, i = \lambda f.f^{-1}, (f \circ g)(x) = f(g(x))$

→ 같은 군(group) 명세를 세 가지 다른 방식으로 해석한 예시.

즉, **같은 서명(Σ)** 이라도 **서로 다른 대수 구조**로 표현될 수 있음을 보여줌.

Order-Sorted Algebra (부분정렬 대수)

- Maude의 정렬 체계는 서브정렬(subsort) 관계를 가질 수 있다.

예: $NzNat < Nat < Int$

- 이 경우 각 정렬의 도메인은 포함 관계를 만족해야 함:

$$\{1, 2, 3, \dots\} \subseteq \{0, 1, 2, 3, \dots\} \subseteq \{\dots, -2, -1, 0, 1, 2, \dots\}$$

핵심 요약

Maude 명세 = (Σ, E)

→ “이 등식들을 만족하는 대수 구조(모델)”들을 정의한다.

→ 그 중 초기 대수(initial algebra)가 Maude의 표준 의미(실행 의미)다.

7.1.1 Homomorphisms and Isomorphisms

Homomorphism (준동형)

- 정의 :

두 Σ -대수 A, B 사이의 함수족(ϕ)이 준동형(homomorphism)이 되려면,

Σ 의 모든 연산 f에 대해 다음을 만족해야 한다.

$$\phi_s(f_A(a_1, \dots, a_n)) = f_B(\phi_{s_1}(a_1), \dots, \phi_{s_n}(a_n))$$

즉, A에서 연산 후 옮기거나, 옮긴 뒤 B에서 연산해도 결과가 같다.

- 의미 : 구조(연산)를 보존하는 함수 → 대수의 “형태적 일관성” 유지.

Example 7.5 — 자연수 ↔ 문자열 대수 간 준동형

- $\phi_{\text{Nat}}(n) = "a"^n$: 자연수 n을 문자 'a'가 n번 반복된 문자열로 매핑.
- $\phi_{\text{Bool}}(x) = x$: 불리언 값은 항등함수.

확인 결과:

$$\begin{cases} \phi(0) = \varepsilon, \\ \phi(s(n)) = \phi(n) + "a", \\ \phi(m + n) = \phi(m) + \phi(n), \\ \phi(m < n) = t \text{ if } m < n \end{cases}$$

→ 모든 연산이 일관되게 보존됨 → 준동형 성립.

Example 7.6 — 정수 ↔ 양의 유리수 대수 간 준동형

$$\phi : \mathbb{Z} \rightarrow \mathbb{R}_{>0}, \quad \phi(x) = 2^x$$

- $e_{\mathbb{Z}} \mapsto 1,$
 $i_{\mathbb{Z}}(x) = -x \mapsto 1/2^x,$
 $x + y \mapsto 2^x * 2^y$
 \rightarrow 모든 군 연산 보존 \rightarrow 준동형.

Example 7.7 — 자연수 \leftrightarrow 짝수 대수 간 준동형

$$\phi(n) = 2n$$

- $0 \mapsto 0,$
 $s(n) \mapsto s(2n) = 2n + 2,$
 $m + n \mapsto 2m + 2n$
 \rightarrow 연산 보존 \rightarrow 준동형.

또한 ϕ 는

- 전사(surjective): 모든 짝수는 어떤 n 에서 $2n$.
- 단사(injective): $m \neq n \Rightarrow 2m \neq 2n.$
 \rightarrow 따라서 동형(isomorphism) 도 성립.

Example 7.8 — 준동형이 존재하지 않는 경우

- $*$ $\rightarrow \mathbb{N}$ 사이에는
 $\text{sign}(\text{NAT-ADD})$ -준동형이 존재하지 않음.
(연산 정의 충돌 \rightarrow 모순 발생.)

Example 7.9 — 준동형이 여러 개 존재하는 경우

- $A = \{1, 2\}, B = \{1, 3\},$ 상수 $a_A = 1, a_B = 1$
- 조건: $\phi(1) = 1$ 이면 성립.
- 가능한 두 준동형:
 $\phi_1 = \{1 \rightarrow 1, 2 \rightarrow 3\}, \phi_2 = \{1 \rightarrow 1, 2 \rightarrow 1\}$

Isomorphism (동형) 정의

$$\phi : A \rightarrow B \text{ 가 동형} \iff \phi \text{ 가 준동형이며, 전사} + \text{단사}$$

- 역함수 ϕ^{-1} 또한 준동형일 때
 A 와 B 는 isomorphic (동형) 대수라 한다.
- 즉, 구조가 완전히 같고 표현만 다르다.

- 전사(surjective) : 모든 결과가 누군가에게서 나옴
- 단사(injective) : 서로 다른 입력은 다른 결과로 감

Example 7.10~7.12 — 동형과 비동형 사례

예제	내용	결론
7.10	$\phi(n) = 2n: \mathbb{N} \leftrightarrow \mathcal{E}$ (짝수 대수)	동형
7.11	$\mathbb{N} \leftrightarrow \text{bits}$	동형
7.12	$\mathbb{N} \leftrightarrow \mathbb{N}_s$	비동형 (단사·전사 불가능)

동형 대수는 구조적으로 동일하며, 원소 표현만 다름.

(예: $0,1,2,3,\dots \equiv 0,2,4,6,\dots \equiv 0,1,0,1,1,\dots$)

Example 7.13 — $\mathbb{N} \leftrightarrow \mathbb{N}_\perp$ (비동형)

$$\phi(0) = 0, \quad \phi(n+1) = \phi(n) + 1$$

→ 가능한 유일한 함수는 항등함수, 하지만 \perp 에는 도달 불가능 → 전사X → 비동형.

7.1.2 Term Algebras

Term Algebra

- signature(Σ)과 변수 집합(X)이 주어졌을 때, Σ 의 기호들로 만들어지는 **모든 식(term)**들의 대수.
- 즉, “기호로만 표현된 식들의 집합” → 실제 계산이 아니라 **기호적 구조** 자체를 다룸.
 - ex : $0, s(0), s(s(0)), s(0 + s(0))$
- 변수가 없을 경우(즉, $X = \emptyset$), 이를 **Ground Term Algebra**라 부른다.

Example 7.14

- $\varphi: \text{term algebra} \rightarrow \text{자연수 대수}$
 - 각 term이 나타내는 실제 수로 해석하는 함수.
 - 조건:

$$\varphi(0) = 0, \quad \varphi(s(t)) = \varphi(t) + 1, \quad \varphi(t_1 + t_2) = \varphi(t_1) + \varphi(t_2)$$

- 즉, 기호적 식을 실제 숫자로 바꿔주는 **준동형(homomorphism)**.
- 반대 방향(자연수 \rightarrow term algebra) 준동형은 **존재하지 않음**.
 - 이유: 자연수에선 0과 0+0이 같지만, term algebra에선 0과 0+0은 서로 **다른 식(term)**이기 때문.

→ 기호적 표현을 실제 의미(숫자)로 해석하는 과정을 보여주기 위한 예제

Example 7.15

- 서명 $\Sigma_0, s = \{ 0 : \rightarrow \text{Nat}, s : \text{Nat} \rightarrow \text{Nat} \}$
- Σ_0, s 대수 $T\Sigma_0, s$ 와 자연수 대수 N 은 **동형(isomorphic)** 구조를 가짐.
(단, 덧셈 연산을 무시했을 때 구조가 동일함.)

7.2 (Σ, E) -Models: (Σ, E) -Algebras

개념 요약

- (Σ, E) -algebra** = 주어진 서명 Σ (기호들의 틀)과 등식 집합 E (규칙들)을 모두 만족하는 수학적 모델.
- 즉, E 에 들어 있는 모든 등식이 A 안에서 항상 참인 구조.

형식적 정의

- 변수 집합 X 이 있을 때, 변수 할당 $\sigma : X \rightarrow A$ 는 각 변수에 값을 넣는 것.
(예: $\sigma(M)=2, \sigma(N)=3$)
- 이걸 식 전체로 확장한 게 ($\sigma^* : T_\Sigma(X) \rightarrow A$).
→ 즉, "기호로 된 식 전체를 실제 값으로 계산하는 함수".

만족 조건

대수 A 가 등식 $(\forall X); t = t'$ 를 만족한다는 건 모든 변수 할당 σ 에 대해 $\sigma^*(t) = \sigma^*(t')$ 이 항상 성립하는 경우야.
→ 즉, "변수에 어떤 숫자를 넣어도 좌우항이 같다."

표기 요약

표기	의미
$A \models (\forall X); t = t'$	대수 A 가 등식 $t=t'$ 를 만족
$A \models E$	A 가 등식 집합 E 전체를 만족
$\text{Alg}(\Sigma, E)$	E 를 만족하는 모든 Σ -대수들의 집합

Example 7.16 — N 은 NAT-ADD 대수임

- NAT-ADD의 등식:
 - $(0 + M = M)$
 - $(s(M) + N = s(M + N))$

이걸 실제 자연수 대수 N 에서 확인하면,

$$(1) 0 + M = M$$

$$\sigma^*(0 + M) = \sigma^*(0) + \sigma^*(M) = 0 + \sigma^*(M) = \sigma^*(M)$$

항상 참 ✓

$$(2) s(M) + N = s(M + N)$$

$$\sigma^*(s(M) + N) = \sigma^*(s(M)) + \sigma^*(N) = (\sigma^*(M) + 1) + \sigma^*(N) = 1 + (\sigma^*(M) + \sigma^*(N)) = \sigma^*(s(M + N))$$

항상 참 ✓

따라서 \mathbb{N} 은 NAT-ADD의 모든 등식을 만족 →

\mathbb{N} 은 NAT-ADD-algebra.

Example 7.17 — $\mathbf{N_x}$ 는 NAT-ADD 대수가 아님

- $\mathbf{N_x}$ (예제 7.3의 대수)에서는 0이 84로 해석되고, +도 일반적인 덧셈이 아님.
- 예를 들어,

$$\sigma^*(0 + M) = \sigma^*(0) +_{N_x} \sigma^*(M) = 84 + \sigma^*(M) \neq \sigma^*(M)$$

이므로, 첫 번째 등식 ($0 + M = M$)부터 깨짐

- 따라서 $\mathbf{N_x}$ 는 NAT-ADD 명세를 만족하지 않음 → $\mathbf{N_x}$ 는 NAT-ADD-algebra가 아님.



대수(algebra)는 숫자 집합(자연수, 정수, 유리수 등)과 그 위에서 정의된 연산들(+, x, s 등)을 포함하는 수학적 구조임.

그리고 (Σ, E) -algebra란 바로 그 구조(대수)에서 명세 E의 모든 등식이 실제로 참인 경우임.

예시로 보면

서명(Σ)	등식(E)	대수(A)	만족 여부
0, s, +	$0 + M = M$, $s(M) + N = s(M + N)$	\mathbf{N} (자연수)	참 - NAT-ADD 대수
0, s, +	같은 등식	\mathbf{Z} (정수)	참 - NAT-ADD 대수
0, s, +	같은 등식	$\mathbf{N_x}$ (기호적 다른 대수)	불만족 - NAT-ADD 대수 아님

7.2.1 Quotient Algebras

핵심 아이디어

Quotient algebra (몫 대수) = “원래 대수의 원소들을 ‘같다고 보는 기준(\approx)’으로 묶은 새로운 대수”

비유로 쉽게 말하면

예 :

- 원래 대수 \mathbf{A} = 자연수 $\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, \dots\}$
- 우리가 “같다고 본다(\approx)”의 기준을 이렇게 정함 : “3으로 나눈 나머지가 같으면 같은 수로 본다”
- 그럼 아래처럼 묶을 수 있음

동치류(묶음)	실제 원소
[0]	{0, 3, 6, 9, ...}
[1]	{1, 4, 7, 10, ...}
[2]	{2, 5, 8, 11, ...}

이게 바로 $A/\approx = \mathbb{N}_3$ 임. 즉, “3으로 나눈 나머지만 신경 쓰는 대수”임.

이걸 수학적으로는 **몫 대수(quotient algebra)** 라고 부름.

왜 굳이 이렇게 하나?

어떤 경우에는 “값이 완전히 다르더라도 **특정 관점에서는 같다**”라고 보고 싶을 때가 있음.

예를 들어 모듈로 연산(mod 3)에서는 3과 6은 같음.

그럼 자연수 전체를 쓰는 게 아니라 “3으로 나눈 나머지”만 쓰면 됨.

이게 바로 “원래 대수를 같다고 보는 기준(\approx)으로 압축시킨 것”임.

즉, **Quotient Algebra = A/\approx** 는 “원래 대수 A를 \approx 로 묶은 압축판(algebra)”임.

그럼 함수나 연산은?

예를 들어 원래 대수에서 덧셈이 있었다면, 몫 대수에서도 덧셈을 **그대로 유지**하지만 이제는 **동치류끼리 더하는 연산**으로 바뀜.

예를 들어, $[1] + [2] = [1 + 2] = [3] = [0]$ (왜냐하면 $3 \equiv 0 \pmod{3}$)

즉, 이 연산은 “원래 덧셈을 그대로 쓰되, 결과를 \approx 기준으로 다시 묶은 것”.

7.2.2 The Algebra

$$\mathcal{T}_{\Sigma, E}$$

이 장은 “명세 (Σ, E)로부터 가장 기본적이고 직접적인 모델(대수)을 어떻게 만드는가”를 설명하는 핵심임.

즉, **명세의 의미(semantics)**를 수학적으로 구현한 대수를 정의하는 부분임.

핵심 개념 요약

$\mathcal{T}_{\Sigma, E}$ (The Algebra $\mathcal{T}_{\Sigma, E}$) : “명세 (Σ, E) 안에 있는 모든 등식이 실제로 참이 되는, 명세로부터 직접 만들어진 기본적인 대수(structure)”

그걸 만드는 아이디어

Step 1. term algebra $\mathcal{T}\Sigma$

- “기호(Σ)로 만들 수 있는 **모든 식(term)** 들의 집합”

$0, s(0), s(s(0)), 0 + 0, s(0) + 0, s(0 + 0), \dots$

Step 2. 등식 집합 E로 “같다고 보기”

- 명세 E 안에는 이런 규칙들이 들어 있음.

$$0 + M = M$$

$$s(M) + N = s(M + N)$$

- 그럼 이런 식들은 실제로는 **같은 의미**를 가지는 식이 됨.

예를 들어,

$$0 + s(0) = s(0)$$

즉, “기호는 다르지만 의미가 같은 식”이 존재함.

그래서 **이걸 동치로 묶어야함**.

Step 3. “같다고 보는 기준” 정의하기

두 식 t, t' 가 같다고 보는 기준:

$$t =_E t' \quad \text{iff} \quad E \vdash t = t'$$

즉, E의 등식들로부터 **증명할 수 있으면 같은 식**으로 취급.

Step 4. 동치로 묶은 결과가 바로 $\mathcal{A}_{\Sigma, E}$

이제 “E로 같다고 본 식들”을 한 덩어리로 묶으면 그게 바로 **몫 대수 (quotient algebra)**

$$\mathcal{T}_{\Sigma, E} = \mathcal{T}_{\Sigma} / \equiv_E$$

즉, “term algebra를 E로 나눈 (=E로 묶은) 결과”가 **명세의 의미를 직접 표현하는 대수**가 됨.

예시 — Example 7.19 (NAT-ADD)

명세:

$$\text{eq } 0 + M = M .$$

$$\text{eq } s(M) + N = s(M + N) .$$

이걸로 term들을 묶으면

동치류 (한 덩어리)	포함된 term들
[0]	{0, 0+0, 0+(0+0), ...}
[s(0)]	{s(0), 0+s(0), s(0)+0, ...}
[s(s(0))]	{s(s(0)), s(0)+s(0), s(s(0)+0), ...}
...	...

즉, 기호로 표현은 달라도 의미가 같은 모든 식들이 **하나의 원소**가 됨.

결과적으로 $\mathcal{A}_{\Sigma, E}$ 는 “자연수의 구조를 정확히 반영한 대수”가 됨. (0, 1, 2, 3, ...을 각각 [0], [s(0)], [s(s(0))], ...로 표현)

Theorem 7.1

$\mathcal{T}_{\Sigma, E}$ is a (Σ, E) -algebra.

즉, " $\mathcal{T}_{\Sigma, E}$ 는 명세 E 에 포함된 모든 등식을 실제로 만족하는 대수이다."

→ 따라서 (Σ, E) 의 **의도된 모델(intended model)** 역할을 함.

7.2.3 The Normal Form Algebra

우리가 앞에서 만든 건 $\mathcal{T}_{\Sigma, E}$, 즉 " E 로 동치인 식들을 묶은 대수"였음. → 이걸 **이론적 모델**임. (어떤 식들이 같은지 증명하는 가능하지만, 실제 계산은 안 됨.) "이제 그걸 실제 계산 가능한 형태(normal form)로 바꾸자."

Normal form이란?

"식(term)을 E 의 규칙으로 다 줄였을 때 더 이상 줄일 수 없는 형태."

예를 들어 NAT-ADD 명세:

```
eq 0 + M = M .
eq s(M) + N = s(M + N) .
```

이 규칙으로 식을 계속 줄이면?

식	줄임 결과
$0 + s(0)$	$s(0)$
$s(0) + s(0)$	$s(s(0))$
$s(0 + s(0))$	$s(s(0))$

즉, 아무리 복잡하게 써도 결국 "**정규형(normal form)**"은 아래처럼 단순해짐.

$0, s(0), s(s(0)), s(s(s(0))), \dots$

즉, 그냥 자연수 $0, 1, 2, 3, \dots$ 과 완전히 1:1 대응.

정규형(normal form)이 가능한 조건

책에 나온 "terminating" 과 "ground confluent" 조건이 이걸 보장해줌.

용어	뜻	쉽게 말하면
terminating	줄이는 과정이 무한히 반복되지 않음	식 줄이다 보면 언젠가 끝남
ground confluent	줄이는 순서가 달라도 결과는 같음	어떻게 줄이든 결과(정규형)는 유일함

이 두 조건이 있으면 모든 식은 단 하나의 최종 형태(normal form)로 수렴함.

그래서 나온 게 정규형 대수 (Normal Form Algebra)

정의 : 정규형(normal form)들만 원소로 가지는 대수

기호로는

$$\mathcal{C}_{\Sigma, E}$$

라고 쓰고,

원소들은 “E로 줄인 식들”이야.

$$\{t!_E \mid t \in \mathcal{T}_\Sigma\}$$

여기서 $t!_E$ 는

“식 t 를 E로 완전히 줄인 결과”
(즉, normal form)

그럼 연산은 어떻게 정의되냐?

대수에는 연산(+, s, 등)이 있음.

그럼 이제 함수 f (예: +, s)를 정의할 때도 “결과를 다시 줄여서(normal form으로) 저장” 하는 식으로 동작함.

즉,

$$f_{\mathcal{C}_{\Sigma, E}}(t_1, \dots, t_n) = (f(t_1, \dots, t_n))!_E$$

→ “ f 를 적용하고, 그 결과를 정규형으로 줄인다.”

예제 — NAT-ADD

명세:

eq $0 + M = M$.
eq $s(M) + N = s(M + N)$.

이건 **terminating + confluent** 이므로 정상적으로 정규형 대수 만들 수 있음.

그럼 이제 정규형 대수 $\mathcal{C}_{\text{NAT-ADD}}$ 의 원소들은

$\{0, s(0), s(s(0)), s(s(s(0))), \dots\}$

즉, 단순한 형태의 “자연수”들임.

덧셈 연산 정의

이 대수 안에서의 덧셈은 이렇게 정의됨.

$$s^m(0) + s^n(0) = s^{m+n}(0)$$

즉,

“s를 m번 적용한 것 + s를 n번 적용한 것 = s를 (m+n)번 적용한 것”

예를 들어:

$$s(s(0)) + s(0) \rightarrow s(s(s(0)))$$

$$(2 + 1 = 3)$$

요약

구분	의미	예시
Normal Form	E의 규칙으로 다 줄여서 더 이상 줄일 수 없는 식	$s(s(0))$
Σ, E	모든 normal form 들로 만든 대수	$\{0, s(0), s(s(0)), \dots\}$
연산 정의	f를 적용한 뒤 결과를 다시 normal form으로 줄임	$s^m(0) + s^n(0) = s^{m+n}(0)$
NAT-ADD 예시	자연수 덧셈 구조 그대로 복원됨	$2 + 1 = 3$

7.3 Soundness and Completeness of Equational Logic

이 장의 핵심 목표

등식 논리(equational logic)는 모든 모델에서 참인 등식을 논리적으로 증명 가능하게 만드는 시스템임을 보인다.
즉,

$$Alg(\Sigma, E) \models (\forall X) t = t' \quad \text{iff} \quad E \vdash t = t'$$

두 가지 속성의 의미

속성	의미	요약 문장
Soundness (타당성)	틀린 걸 참이라고 증명할 수 없다.	"증명 가능한 것은 실제로 참이다."
Completeness (완전성)	참인 건 반드시 증명할 수 있다.	"모든 참을 논리로 증명할 수 있다."

왜 Soundness는 쉽고 Completeness는 어려운가?

- **Soundness**는 "규칙이 상식적이냐"만 확인하면 됨.
→ 규칙이 거짓을 참으로 만들지 않으면 끝.
- **Completeness**는 "모든 모델에서 참인 것을 다 증명할 수 있느냐"를 보여야 함.
→ 즉, 무한히 많은 모델에서 참인 것을 하나의 논리 시스템으로 커버해야 함.

Completeness 증명의 핵심 아이디어 (버코프 정리 Birkhoff's Theorem)

1. "E로부터 증명 가능한 식들만 같다"고 정의한 특수한 모델 **S**를 만든다.

$$t =_S t' \iff E \vdash t = t'$$

2. 이 모델 **S**가 실제로 (Σ, E) -algebra임을 보인다.
3. 그러면

$$Alg(\Sigma, E) \models (\forall X) t = t' \Rightarrow E \vdash t = t'$$

가 자동으로 따라온다.

→ 즉, "모든 모델에서 참이면, 이 논리로도 증명 가능하다."

Completeness 증명 (핵심 구조)

1 $E \not\vdash t_1 = t_2$ 라고 가정한다.

→ 즉, $t_1 = t_2$ 는 E로부터 증명 불가능하다.

2 그러면 t_1, t_2 는 서로 다른 원소가 된다.

$$[t_1]_{=E} \neq [t_2]_{=E}$$

3 따라서 quotient algebra

$$\mathcal{T}_\Sigma(X) / \equiv_E$$

안에서는 $t_1 \neq t_2$ 이다.

4 그런데 이 algebra는 실제 (Σ, E) -algebra이므로,
"모든 모델에서 참"일 수 없다.

→ 결론: $E \not\vdash t_1 = t_2$ 라면,

$\text{Alg}(\Sigma, E) \not\models (\forall X) t_1 = t_2$ 이다. ✓

(즉, 완전성의 대우 증명)

Soundness 증명 (귀납법으로)

1 $E \vdash t_1 = t_2$ 가 증명되었다면,

어떤 모델 A에서도 $\sigma^*(t_1) = \sigma^*(t_2)$ 임을 보인다.

2 증명의 길이(length)에 대해 귀납한다.

- 기저 단계 (length = 0)
 - Reflexivity: $t_1 = t_2$ 자체가 같은 항이므로 성립.
 - Substitutivity: E 안의 식 $t = t'$ 를 대입해도 모델에서 같음이 유지됨.
- 귀납 단계 (length = n+1)
 - Symmetry: $E \vdash t_2 = t_1$ 이면 $\sigma^*(t_2) = \sigma^*(t_1)$
 - Transitivity / Congruence: 중간 단계가 같으면 전체도 같음.

→ 즉, "증명된 등식은 어떤 모델에서도 항상 참이다." ✓

결론: Birkhoff's Completeness Theorem

정리 7.2 (Birkhoff's Completeness Theorem)

주어진 등식 집합 E 에 대해,
모든 항 $t_1, t_2 \in \mathcal{T}_\Sigma(Y)$ 에 대해

$$Alg(\Sigma, E) \models (\forall Y) t_1 = t_2 \iff E \vdash t_1 = t_2$$

즉,

- “ E 로 증명 가능하다” \leftrightarrow “모든 모델에서 참이다”
- 등식 논리는 완전하고 타당하다.

한 줄 요약 (암기용)

등식 논리는 sound하고 complete하다.

⇒ “틀린 걸 증명할 수 없고, 참인 건 반드시 증명할 수 있다.”

⇒ 이를 보이기 위해 ‘증명 가능한 등식만 같다’고 정의한 모델을 만들어 증명한다.

7.4 Intended Models : Initial Algebras

핵심 개념 한 줄 요약

우리가 의도한 모델은 ‘초기 대수(Initial Algebra)’다.

즉, 같은 명세 (Σ, E) 를 만족하는 여러 모델 중에서도 가장 기본적이고 불필요한 요소(“junk”)가 없는 모델이 의도된 모델이다.

문제의 시작: “진짜 모델은 누구인가?”

하나의 명세 (예: NAT-ADD)를 놓고 보면, 그걸 만족하는 모델이 여러 개 있을 수 있다.

예를 들어 NAT-ADD의 경우:

- 자연수 대수 \mathbb{N}
- 비트(binary) 대수 bits
- 짝수만 있는 대수 \mathcal{E}
- squares, +2, AB ...

등이 모두 (Σ, E) -algebra가 될 수 있음.

그러면 “진짜 우리가 의미한 자연수”는 이 중 어떤 것일까?

답: 초기 대수(Initial Algebra)

Goguen, Thatcher, Wagner, Wright(1975)의 답변:

“명세 (Σ, E) 의 의도된 모델은 (Σ, E) -algebra들 중 초기(initial) algebra이다.”

정의: Initial Algebra

Definition 7.5 A Σ -algebra A is initial in a class \mathbb{A} of Σ -algebras if and only if for each algebra $B \in \mathbb{A}$, there is exactly one Σ -homomorphism from A to B .

즉, 초기 대수 A 는 “다른 모든 대수로 가는 유일한 길”이 존재하는 대수다.

직관적으로는 이렇게 생각하면 됨 : “초기 대수는 다른 모든 모델의 ‘뼈대’가 되는 최소 모델이다.”

성질 1: 초기 대수는 유일하다 (up to isomorphism)

Theorem 7.3

만약 두 초기 대수 A와 B가 존재한다면, 둘은 서로 동형(isomorphic)이다.

즉, 여러 개의 초기 대수가 존재하더라도, 결국 “내용이 같은” 하나의 모델이라고 봐도 무방하다.

성질 2: 항상 하나의 초기 대수가 존재한다

Theorem 7.4

$\mathcal{T}_{\Sigma, E}$ (즉, ground term algebra의 quotient)
는 (Σ, E) -algebra들 중 하나의 초기 대수다.

이게 바로 **의도된 모델(Intended Model)** 이다.

이 모델은 “모든 식이 E로부터 증명 가능한 만큼만 동일하다고 보는” 가장 단순하고 근본적인 대수다.

예시: NAT-ADD

- NAT-ADD의 초기 대수는 $\mathcal{T}_{NAT-ADD}$ 이다.
- 이진 “0, s(0), s(s(0)), ...” 같은 모든 기본 식(ground term) 으로 구성된다.
- 자연수 대수 \mathbb{N} 과도 동형(isomorphic) 이다.
→ 즉, 우리가 아는 자연수 모델이 곧 초기 대수.

$$\phi([s(s(\cdots s(0) \cdots))]) = n$$

이 함수 ϕ 는 (Σ, E) -동형사상(homomorphism)이며,
자연수 모델과 $\mathcal{T}_{NAT-ADD}$ 가 같은 구조임을 보인다.

성질 3: 정상형 대수 (Normal Form Algebra)

Theorem 7.5

만약 E가 **terminating**(종료) 하고 **confluent**(일관) 하다면,
Normal Form Algebra $\mathcal{C}_{\Sigma, E}$ 는
 $\mathcal{T}_{\Sigma, E}$ 와 동형이며,
따라서 초기 대수다.

즉, 우리가 실제로 Maude에서 계산해서 얻는 **정상형(normal form)** 들로 이루어진 대수가
바로 의도된 모델과 동일하다.

초기 대수의 두 가지 중요한 특성

성질	의미
No Junk	대수 안에 “불필요한 원소”가 없음. 즉, ground term의 해석으로 얻을 수 없는 원소가 없다.
No Confusion	E로부터 동등하다고 증명되지 않은 두 항은 같다고 취급하지 않는다.

이 두 성질을 만족하면 그 대수는 초기 대수와 **동형(isomorphic)** 이 된다.

예시로 이해하기

예를 들어 \mathbb{N}_3 (mod 3 정수 대수)에서는
 $s(s(s(0)))$ 와 0 이 같은 값으로 해석된다.
 하지만 NAT-ADD에서는
 $s(s(s(0))) \neq 0$ 임을 증명할 수 있다.
 → 따라서 \mathbb{N}_3 는 no-confusion을 위반함.
 → 즉, 의도된 모델이 아님.

결론 요약

개념	설명
Intended Model	우리가 명세로 정의한 "진짜" 모델
Initial Algebra	모든 모델로 가는 유일한 homomorphism이 존재하는 최소 모델
No Junk / No Confusion	초기 대수의 특징 (불필요한 원소 없음, 잘못된 식 동일시 안 함)
결론	$\mathcal{T}_{\Sigma, E}$ (또는 $\mathcal{C}_{\Sigma, E}$) 가 바로 intended model이다.

요약 한 줄

초기 대수(Initial Algebra) 는 명세 (Σ, E) 의 **의도된 모델**이며, "불필요한 원소(No Junk)"도 없고 "잘못된 동일시(No Confusion)"도 없는 가장 단순하고 본질적인 모델이다.

7.5 Empty Sorts and Many-Sorted Equational Logic

핵심 요약

어떤 sort(정렬)에도 ground term이 하나도 없다면, 기존의 등식 논리(equational logic)를 그대로 확장해서 쓰면 논리가 무너지다(**unsound**).

기본 아이디어

"unsorted"에서 "many-sorted" (즉, 여러 sort가 있는 경우)로 확장하는 건 보통 쉽다.

단, 모든 **sort**가 적어도 하나의 **ground term**을 가지고 있다면 괜찮다.

하지만 어떤 sort가 **ground term**이 전혀 없을 때는 문제가 생긴다.

이게 바로 **Empty Sort** 문제다.

예제 (Example 7.23)

```
fmod EMPTY-SORT is including BOOL .
sort Empty .
op f : Empty → Bool .
var X : Empty .
eq f(X) = true .
```

```
eq f(X) = false .
endfm
```

설명 :

- `Empty` sort에는 **ground term**이 없음 (즉, `Empty` 타입의 실제 값이 존재하지 않음)
- 하지만 `f : Empty → Bool` 은 정의됨
- 식 `f(X) = true` 와 `f(X) = false` 두 개가 존재함

모델 해석 (Model A)

이 명세의 한 모델 A를 보자:

- $A_{Empty} = \emptyset$ (`Empty` sort는 공집합)
- $A_{Bool} = \{t, f\}$

함수 f 는 다음과 같이 정의할 수 있다:

$$f(e) = t \quad \forall e \in \emptyset$$

즉, 입력이 없으니까 아무 문제 없이 참으로 정의 가능함.

이 A는 명세의 정상적인 모델이다.

왜냐면:

- “모든 $X \in \emptyset$ 에 대해 $f(X) = false$ ”는 자동으로 참이다.
(공집합에 대한 전칭명제는 vacuously true)
- 따라서 두 등식 $f(X) = true$ 와 $f(X) = false$ 모두 만족함.
(둘 다 빈 집합에 대해 참이 되므로 충돌 없음)

그런데 문제가 생김

이제 **unsorted equational logic**을 그대로 확장해서 쓰면,
논리적으로 다음을 증명할 수 있게 되어버림:

$$EMPTY-SORT \vdash true = false$$

하지만 실제 모델 A에서는

`t` 와 `f` 가 서로 다르기 때문에 이건 거짓이다.

☞ 즉, 논리가 **unsound**(비타당) 해짐.

(논리적으로 증명할 수 있는 게 실제 모델에서는 성립하지 않음)

왜 이런 일이 생기나?

`Empty` sort 안에는 변수를 대입할 **ground term**이 없음.

그래서 식에 나오는 “ $\forall X : Empty$ ”는 실제로는 아무 것도 검증하지 않는데, 논리 체계는 “모든 X 에 대해 참이다”라고 착각한다.

→ 즉, “공집합에 대해 모든 게 참으로 간주되는 vacuous truth” 때문에 모순된 등식(`true = false`)까지도 증명 가능한 상황이 생긴다.

해결책

Meseguer & Goguen이 **sound**하고 **complete**한 **many-sorted equational logic**을 새로 정의함.


이 논리에서는 변수를 정렬별로 조심스럽게 다루기 때문에,

다음은 증명 가능하지만,

$$EMPTY-SORT \vdash (\forall X : Empty) true = false$$

이건 증명 불가능하다:

$$EMPTY-SORT \not\vdash true = false$$

즉, 그들의 체계에서는 soundness가 유지된다. 

요약 정리

개념	설명
Empty Sort	ground term이 하나도 없는 sort
문제점	기존 equational logic을 그대로 확장하면 unsound해짐
원인	공집합에 대해 전칭명제가 “항상 참”이 되어버려 잘못된 등식(true=false)도 증명 가능
해결책	Meseguer & Goguen의 many-sorted equational logic — 변수를 정렬별로 엄밀히 다룸
결론	Empty sort를 다룰 땐 기존 논리를 그대로 쓰면 안 된다. Soundness를 깨뜨림.

한 줄 요약

Empty Sort(비어있는 정렬) 때문에 “공집합에서는 뭐든 참”이라는 논리적 함정이 생겨 기존 등식 논리를 그대로 확장하면 **unsound(비타당)** 해진다. Meseguer & Goguen은 이를 수정해 **sound**하고 **complete**한 **many-sorted equational logic**을 제안했다.