17. Real-Time and Probabilistic Systems

17.1 Real-Time Systems

- 실시간 시스템 : **이벤트 간의 시간(duration)**이 시스템의 기능에 영향을 미치는 시스템
- 이전의 단순 분산 시스템 가정(메시지 유실 없음 등)과 달리, 많은 실제 시스템에서는 **시간 요소를 무시할 수 없음**
- 시간 정보가 필요한 주요 이유 :
  - **결함 허용(Fault tolerance)** : 메시지 손실 및 노드 장애 판단에 시간 기반 추론이 필수
  - **성능 향상** : 시간은 분산 알고리즘 성능 최적화에 중요한 변수
  - **임베디드 시스템** : 물리 환경과 상호작용하며 시간 제약이 존재
  - **시간 제약 속성 검증** : "결국(eventually)"이 아닌 "정해진 시간 이내"를 보장해야 함
    - 예: *충돌 후 10ms 이내에 에어백 전개*
- Real-Time Maude는 이러한 실시간 시스템 모델링과 분석을 지원하는 도구

17.1.1 Specifying Real-Time Systems in Rewriting Logic

두 종류의 Rewrite Rule

종류	의미	시간 변화 존재?
Instantaneous Rule	즉시 일어나는 사건 (배터리 나감, 충돌 감지 등)	X
Tick Rule	시간 흐름을 모델링 (1초 흐름 등)	O

- Instantaneous = 0시간
- Tick = 시간 τ 만큼 증가

시간 적용 방식

- Maude는 전체 시스템을 아래처럼 감쌈.

  ```
  { State } in time t
  ```

  - **t : 시작 후 누적된 전체 시간**
- tick rule:

  ```
  {t} → {t'} in time τ
  ```

  - 시간이 τ만큼 흐르면서 상태는 t에서 t'로 변화
  - 중요: 시간은 **모든 객체에 동등하게 적용**

---

mte & timeEffect = Real-Time 핵심 두 함수

함수	의미
timeEffect(conf, τ)	τ 시간 흐른 뒤 각 객체의 상태 변화
mte(conf)	더 이상 기다리기 전에 반드시 발생해야 하는 사건까지의 최대 대기시간

- Tick rule 실행 조건:

  ```
  τ ≤ mte(conf)
  ```

  = 너무 오래 기다릴 수는 없음 = **중요 이벤트 전에 시간 멈춤**

예제: 시계 모델

- running → 시간이 흐름
- time = 12 → 즉시 0으로 점프
- 언제든지 고장남
- 시간 tick:

  ```
  if time < 12 → time = time + 1
  else → time = 0
  ```

메시지 지연 모델링

- 메시지에 남은 지연 시간 t를 붙여 표현:

  ```
  dly(msg, t)
  ```

- 시간 경과 시:

  ```
  dly(msg, t) → dly(msg, t - τ)
  ```

- t = 0 되면 **읽을 수 있는 메시지**

Unbounded → Infinite → Search 못함

- 실시간 시스템은 시간이 무한히 흐를 수 있기 때문에 검색(search)이 끝없이 돌 위험 → **실행시간 추상화 필요**
- 예: system clock 제거

  ```
  eq {t} in time T = {t}
  ```

  → 시간 정보를 떼고 **상태만 비교**

요약

- 실시간 시스템은 **시간 제약이 기능에 영향을 준다**
- Tick rule은 **시간 흐름을 모델링**
- `mte`는 **최대 대기시간**, `timeEffect`는 **시간 경과 후 상태 변환**
- 시간은 시스템 전체에 **동일하게 적용**

- 시간의 무한 증가 → 검색 종료 어려움 → **시간 추상화 기법 필요**

17.1.2 Timed Temporal Logics

- 실시간 시스템 요구사항은 종종 **시간 제약이 포함된 속성(Timed properties)** 형태로 표현된다.
- 예시:
  - “충돌이 감지된 후 **10ms 이내에** 에어백이 전개되어야 한다.”
  - “수술 중 호흡장비는 **10분 동안 단 한 번**, 그리고 **최대 2초 이내**로만 일시정지될 수 있다.”
- 이를 명확하게 표현하기 위해 템포럴 로직을 확장한 **Metric Temporal Logic (MTL, 시간 구간 기반 논리)** 을 사용한다.
- MTL에서는
  - □(항상), ◇(언젠가), U(까지) 연산자에 시간 구간을 붙여 명세 가능
- 예시 공식화를 통해 이해

자연어 요구사항	공식 표현
충돌 후 10ms 이내 에어백 전개	□≥0 (crash → ◇≤10ms airbag)
호흡장비 정지 제한 요구사항	□≥0 (paused → ◇≤2sec (□≤10min ¬paused))

- 즉, **언제까지**, **얼마 동안**, **몇 번**과 같은 요구를 정확하게 논리로서 명시/검증 가능하게 만든다.

17.1.3 Real-Time Maude

- Real-Time Maude는 **실시간 시스템을 명세/시뮬레이션/검증하기 위한 도구**이다.
- 기능:
  - 특정 시점까지 시스템 시뮬레이션
  - 특정 시간 구간 내에서 도달 가능한 패턴 탐색
  - **시간 제약을 포함한 LTL 모델체킹(timed temporal logic model checking)** 지원
- 적용 분야:
  - 무선 센서 네트워크
  - 클라우드 스토리지 시스템
  - 산업용 모델링 언어 (예: AADL, Ptolemy II)
  - CPS 시스템 안전성 검증
- 추가 특징:
  - **Randomized simulation**을 지원하여 성능(performance) 분석에도 활용 가능

17.2 Probabilistic Systems

- 일반적인 재기 규칙 $t \to t'$ 은 **"t에서 t'에 도달할 수 있음"** 을 의미할 뿐, 그 일이 **얼마나 자주** 발생하는지(확률)는 표현하지 않는다.
- 그러나 실세계 시스템은 종종 확률을 고려해야 한다:

확률을 반드시 모델링해야 하는 이유

이유	설명
확률적 결과 분석 필요	"카지노에서 $200 이상 딸 확률은?"
안전성 인증 기준에 필요	항공기 시스템: "10억 비행시간당 1번 이하 사고"
성능/품질 예측	응답률, 성공률, 가용성, 평균 지연 등
시스템 자체가 확률적임	메시지 손실, 컴포넌트 고장, 랜덤화 알고리즘

- 예: QuickSort → pivot을 랜덤하게 선택할 경우 평균 성능 향상

모델체킹 관점에서 확률 필요성

- 엄밀한(완전한) 모델체킹은 상태공간이 커지면 **실패**할 수 있다.
- 그래서 대신, **통계적 모델체킹(Statistical Model Checking, SMC)**
  - 충분히 많은 **랜덤 시뮬레이션**을 통해
  - 높은 확신 수준(confidence)으로 결과 평가
- 장점
  - 모든 reachable state를 저장할 필요 없음 → 메모리 효율
  - 병렬화 용이 → 대규모 시스템 분석 가능

재기논리와 확률

- 확률적 시스템도 **Probabilistic Rewriting Logic**으로 명세 가능
- 하지만 실행을 위해서는 **통계적 모델체커와 연결해야 함**
- 예: VeStA / PVeStA / MultiVeStA
- 이 도구들은 다음을 추정한다.
  - 실행 결과의 **기대값(average value)**
  - 특정 속성을 만족할 **확률(probability)**
  - 예:
    - 블랙잭 테이블에서 남는 기대 금액
    - $200 이상 딸 확률

실제 연구/산업 적용 예시

- DoS 공격 방어 메커니즘 효율 평가
- Apache Cassandra 데이터스토어 성능 분석 및 최적화
- 실제 코드 비교 기반 검증
- 최신 무선 센서 네트워크 알고리즘 평가

17.2.1 Probabilistic Rewrite Theories

확률적 재기 규칙이란?

- 일반 재기 규칙은 "A → B 가능함" (확률 정보 없음)
- 하지만 실제 시스템에서는 "A가 B로 바뀔 확률이 30%" 같은 구체적인 확률 모델링이 필요
- 그래서 확률적 재기 규칙은 다음과 같은 형태를 가진다:

```
t → t' with probability y := dist(x1, ..., xn) if cond
```

  - `dist`는 확률 분포(bernoulli, uniform 등)
  - 각 실행마다 결과가 확률적으로 **샘플링**
  - 동일한 규칙이라도 **상태에 따라 확률 달라질 수 있음**

예시 (Example 17.5)

```
a → b 확률 30%
a → c 확률 70%

c → d 확률 40%
c → e 확률 60%
```

- **한 상태에서 여러 확률적 전이가 존재할 수 있음**

생존/사망 모델 예시 (Example 17.6)

```
나이가 X일 때
사망 확률 = X^4 / 120^4   (예시 값)
```

- Bernoulli 분포 사용
  - true 나오면 죽음
  - false 나오면 한 살 더 먹음
- 확률이 **상태 값(age)**에 따라 달라짐

블랙잭 예시 (Example 17.7)

- 카드 뽑기 이벤트는 uniform 분포

```
모든 남은 카드가 동일 확률로 선택됨
```

- 게임 규칙 조건에 따라 hit 가능 여부 결정 (예: 최소 합 15 이상이면 hit 불가)

실행을 위해 일반 재기 이론으로 변환 필요

- 현재 Maude는 **확률적 규칙을 직접 실행 못 함**
- 그래서 확률적 규칙 → 일반 규칙 + random 함수

- Maude의 `random(counter)` → 난수 생성
- 이를 이용해 Bernoulli나 uniform을 구현하여 시뮬레이션 가능

순수 확률적 모델 얻기

- 정확한 통계 검증을 위해 확률 외 불확정성(nondeterminism) 제거 필요
- 방법:
  - 메시지 도착 시간에 **연속 확률 분포** 적용
  - 두 메시지가 **동시에** 도착할 가능성 → 0
  - 어떤 이벤트가 발생할지 명확해짐
- 안전하게 확률만으로 시스템의 모든 불확실성을 표현 가능

17.2.2 Probablistic Temporal Logics

- 개념
  - 기존 **시간 논리(temporal logic)**에 **확률** 요소를 추가한 논리.
  - 즉, "**언제** 발생하는가 + **얼마나 높은 확률로** 발생하는가"를 함께 검증할 수 있음.
- 예시
  - $\mathscr{P} \geq 0.9$ (□ (crash → ◇≤10ms airbag))
    - 사고 발생 후 **10ms 이내**에 에어백이 **90% 이상 확률로** 전개되어야 함.
    - 실제 안전 시스템 요구사항을 더 현실적으로 표현 가능

Probabilistic Model Checker

- 모델이 위와 같은 "확률적 시간 요구사항"을 만족하는지 체크해주는 도구들 존재
- 두 종류가 있음:

종류	특징	장점	단점
정확한 확률 모델 체크	실제 확률 값을 계산	항상 정답 보장	**상태 폭발 문제**로 큰 시스템에 적용 어려움
통계적 모델 체크 (Statistical Model Checking)	랜덤 시뮬레이션 기반	확장성 좋음, 병렬화 쉬움	확률적으로 "맞을 가능성이 높다" 수준의 결과

비교 예시

- Property Ψ: 상태 a에서 시작할 때, 상태 e에 도달할 확률 ≥ 0.41
  - 실제 확률: **42%**
  - 정확 모델 체크 → 항상 Ψ 성립이라고 결론
  - 통계 모델 체크 → 샘플링 운 나쁘면 Ψ 불성립이라고 말할 가능성 있음
- 즉,
  - 정확 모델 체크 = 확신은 높지만 큰 시스템에 취약
  - 통계 모델 체크 = 확장성 높지만 약간의 오차 가능

17.2.3 PVeStA Analysis

- **PVeStA**는 *확률적 시스템의 성능/품질을 통계적으로 평가*하는 도구 모음이다.
- Maude 모델과 연동하여 **무작위 시뮬레이션을 수행**하고, 그 결과를 기반으로 **확률적 속성**을 추정한다.
- 주요 평가 대상
  1. **기대값(Expected value)**
     - 예: blackjack에서 최종 남은 돈의 평균
  2. **확률(Expected probability)**
     - 예: 목표 금액 이상 벌고 나올 확률
  - 즉, 다음과 같은 속성 분석이 모두 가능: "어떤 속성이 p 이상의 확률로 만족되는가?"

PVeStA가 필요한 이유

- 전통적인 확률 모델체킹은 **모든 상태 탐색 → 폭발적**
- PVeStA는 **통계적 샘플링 기반 → 대규모 시스템에도 적용 가능**
- 신뢰도(Confidence)와 오차 경계(δ)를 사용하여 필요한 샘플 수를 자동 조정
  - 예: 99% 신뢰도로 ±1 범위의 오차 내에서 평균 값을 추정한다.

예시 결과

1. **$1000 → 20번 blackjack 플레이**

- 기대 잔액: **$876**
- 20판 후 **$1200 이상 될 확률: 31%**

2. 딜러 규칙 변경 시 기대 잔액: **$826**
3. 신생아 기대수명(모델 기반): **58년**
4. 65세까지 생존 확률: **34%**

→ 결과는 여러 번의 PVeStA 실행 평균 값으로 도출됨

한 줄 요약

- “PVeStA = 확률적 성질을 정확도와 신뢰도 관점에서 추정할 수 있는 대규모 확률 시스템용 통계적 모델 검증 도구”