Representing Embedded System Sequence Diagrams as a Formal Language

Beth Latronico (beth@cmu.edu)
Philip Koopman (koopman@cmu.edu)
Carnegie Mellon University
Electrical and Computer Engineering Department

Electrical & Computer ENGINEERING

Overview

- Motivation and problem definition
- Embedded system example
- Generic solution
- Solution applied to example
- Additional information and conclusions

Motivation

- Statechart synthesis from sequence diagrams (SDs)
  - Benefits
    - Enhanced traceability (specification <-> design)
    - Algorithms for synthesis have been previously proposed
- What are the ramifications of specification omissions and conflicts?
  - Statecharts may contain unwanted non-determinism
  - Informal resolutions may be inadequate
    - Add information: Exhaustive annotation often infeasible
    - Locate non-determinism: Manual inspection affords opportunity for error (Pairwise comparison of SDs)

Solution Properties

- Key observation: *Missing information* in SDs may lead to unwanted *non-determinism*
  - How can we minimize information annotation effort?
  - How can we devise a consistent screening method for non-determinism that can verify removal?
- Research contribution
  - Treat SDs as a formal grammar
    - Attack errors at specification level – reduce lifecycle costs
  - Analyze this notation for non-determinism
    - Annotate diagrams at specific locations
    - Verify removal of non-determinism
    - Detection could be automated!

Define Problem Space

- What makes a system more difficult to specify?
  - Combined characteristics (typical of embedded):
    - Multiple initial start states (e.g. radio on, radio off, CD in)
    - Same user action invokes different response (e.g. radio clock set)
    - Timing dependencies (e.g. hold time for radio button)
- What information is typically added to SDs?
  - Regardless of representation format, designers tend to add information about:
    - State – Present behavior depends on past
    - Data – Behavior depends on value of a variable
    - Time – Behavior depends on properties such as latency, duration, or absolute time

Define Solution Space

- What will the set of SDs look like?
  - Individual diagram information
    - Standard Sequence Diagrams (objects and messages)
    - Additional information (state, data, time) as needed
    - Formal grammar analysis here
  - Composition information
    - Based on high-level Message Sequence Chart
    - (Not in UML 1.3 standard – coming soon?)
- How is the grammar defined?
  - Deterministic - one unique response set per unique message set
    - Leverage compiler theory

Motivational Example – Car Radio Controller

Here is a (small) typical car radio controller scenario.

Select actors, *messages* and **objects** for SD.

1. The driver presses a station **button**.

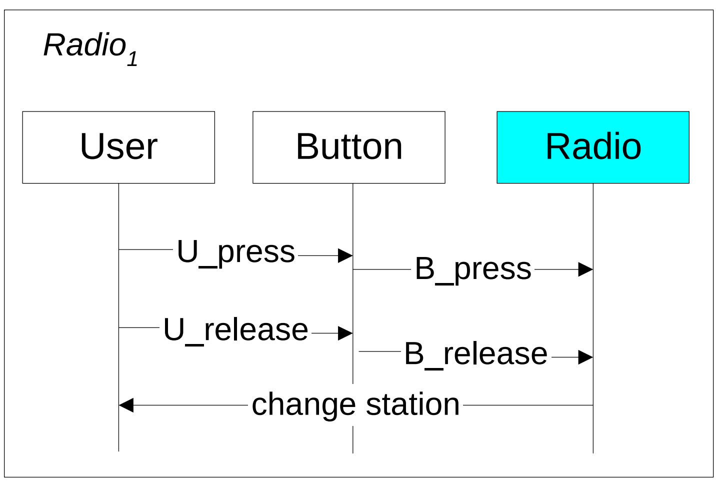
   1.A. If the driver *holds* the **button**, the station is *set*.

   1.B Otherwise, the **radio** should *change* stations.

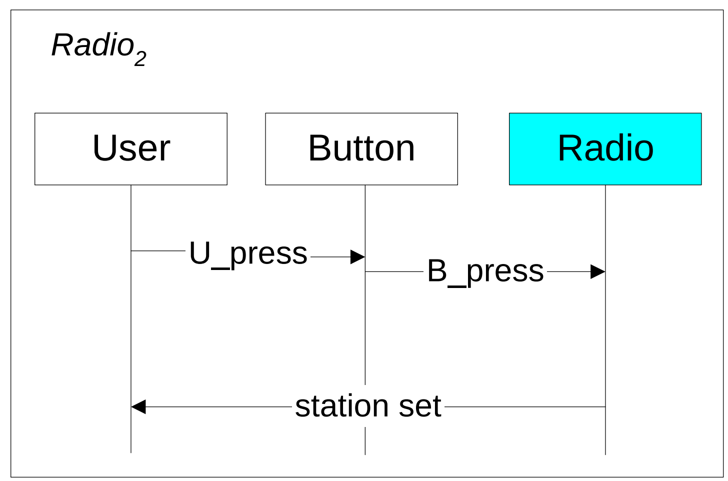
Note that 1.A. doesn't tell how long the button should be held. What are the ramifications?

Objects and Messages

- Embedded example – Car radio controller
  - Design of Radio object
    - Simple example to illustrate point
  - Two standard SDs – change station, station set



Radio$_1$: Change station



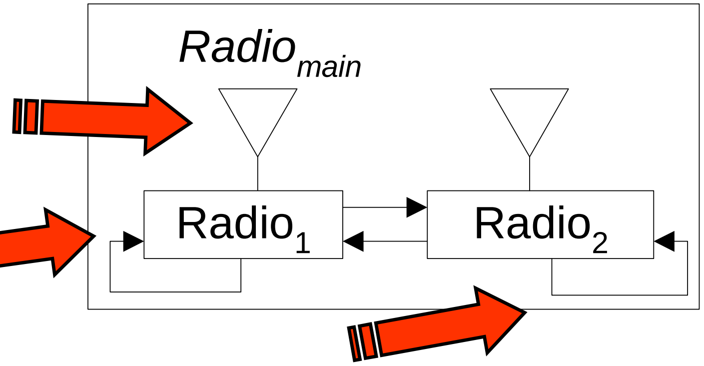
Radio$_2$: Station set

Diagram Composition

- High-level message sequence chart (MSCs)
  - (Established by the MSC community)
  - Constituent diagrams (here, Radio1 and Radio2)
  - Possible initial choices (indicated by triangles)
  - Allowed order of execution

Initially, either Radio$_1$ or Radio$_2$ can be executed.

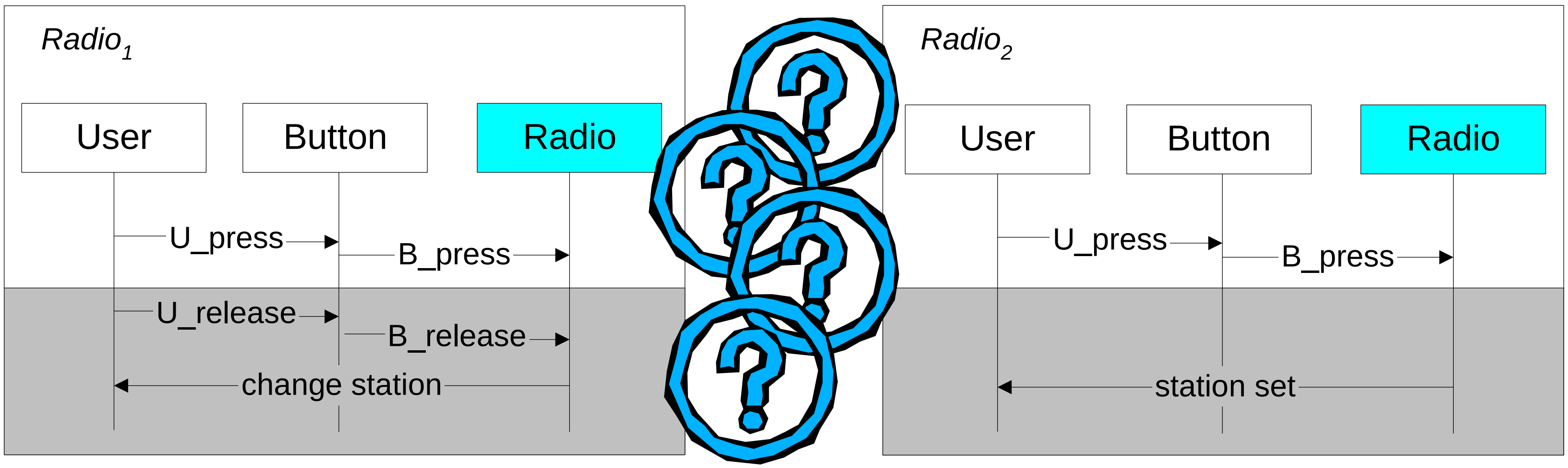
After Radio$_1$, either Radio$_1$ or Radio$_2$ can be executed.



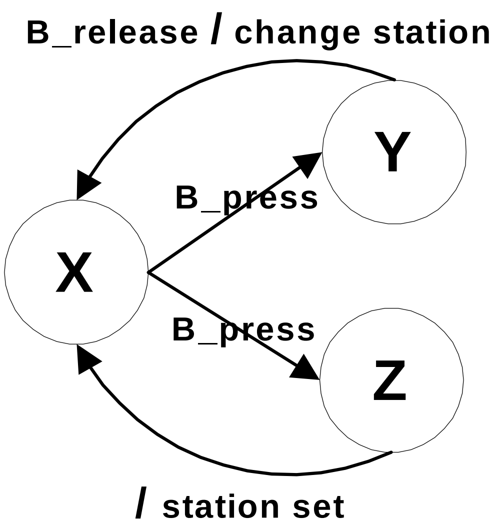
After Radio$_2$, either Radio$_1$ or Radio$_2$ can be executed.

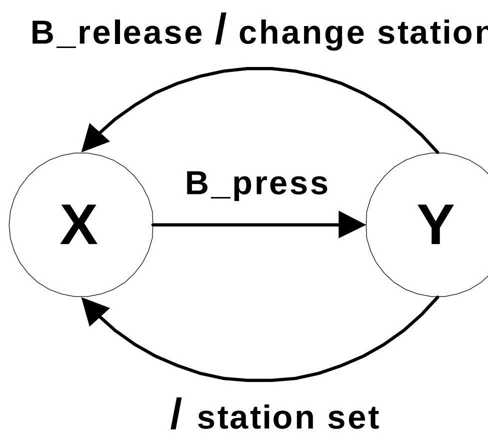
Non-determinism Arises

Consider only the Radio object. A ‘B_press’ message arrives…

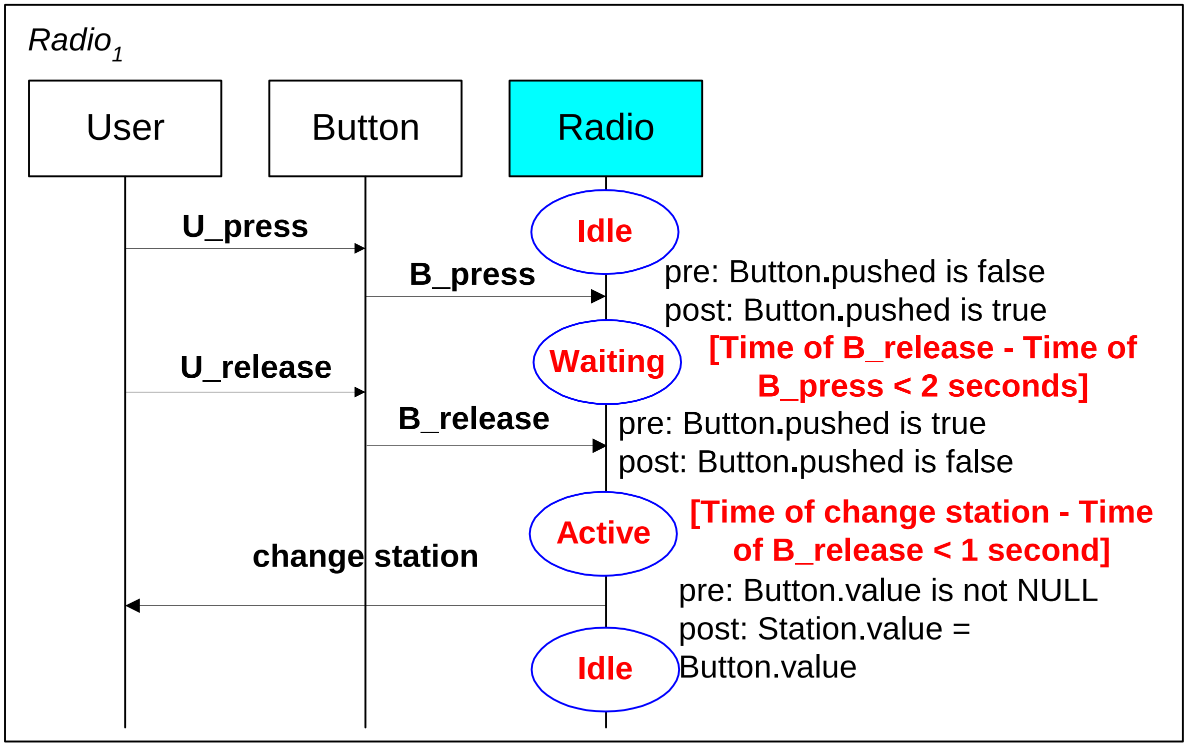


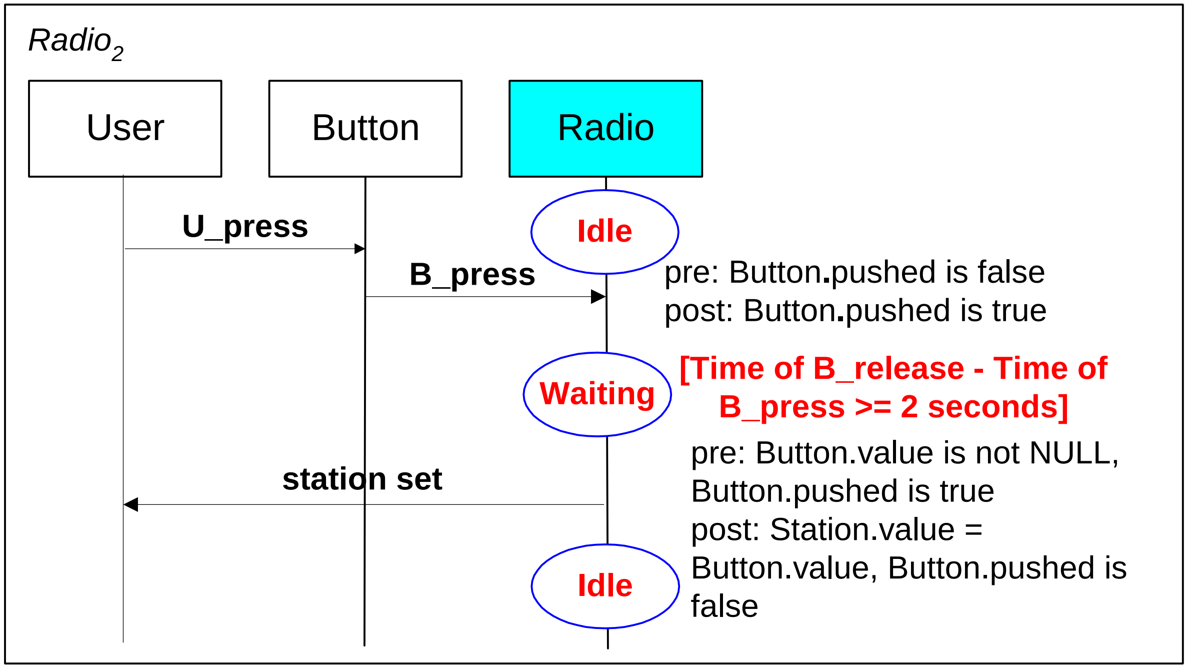
Two possible statecharts for Radio object: (both non-deterministic!)





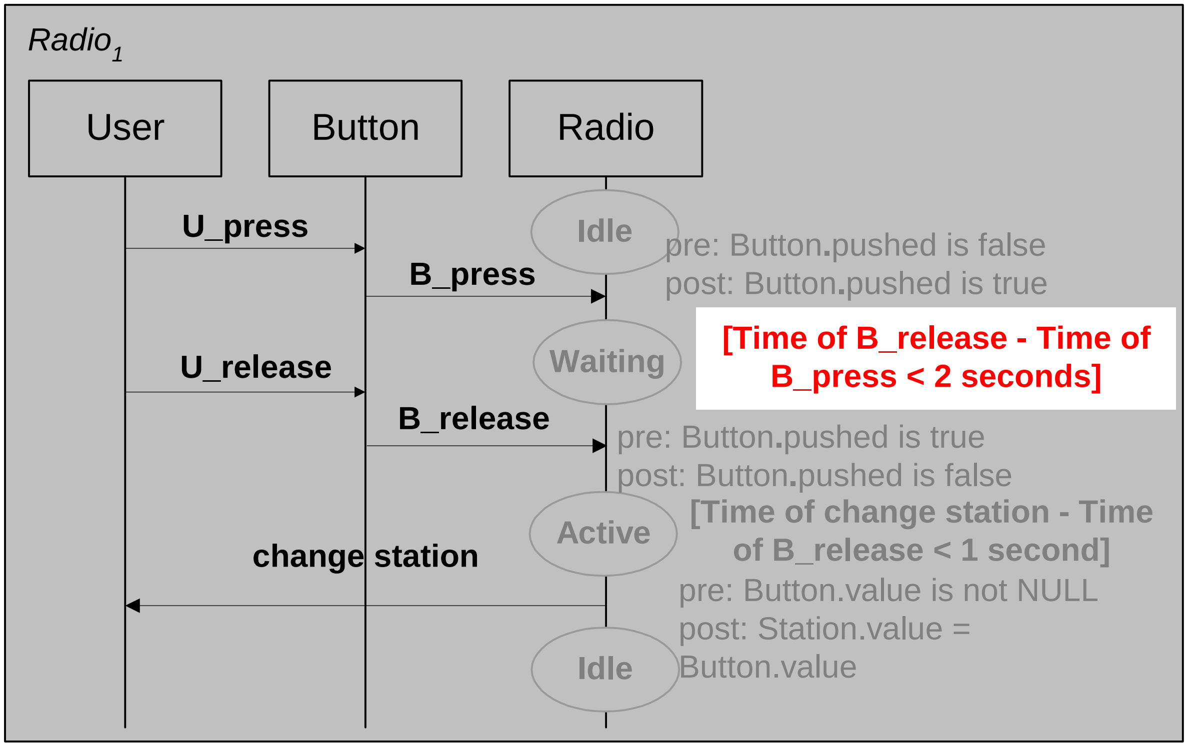
…We need more information.

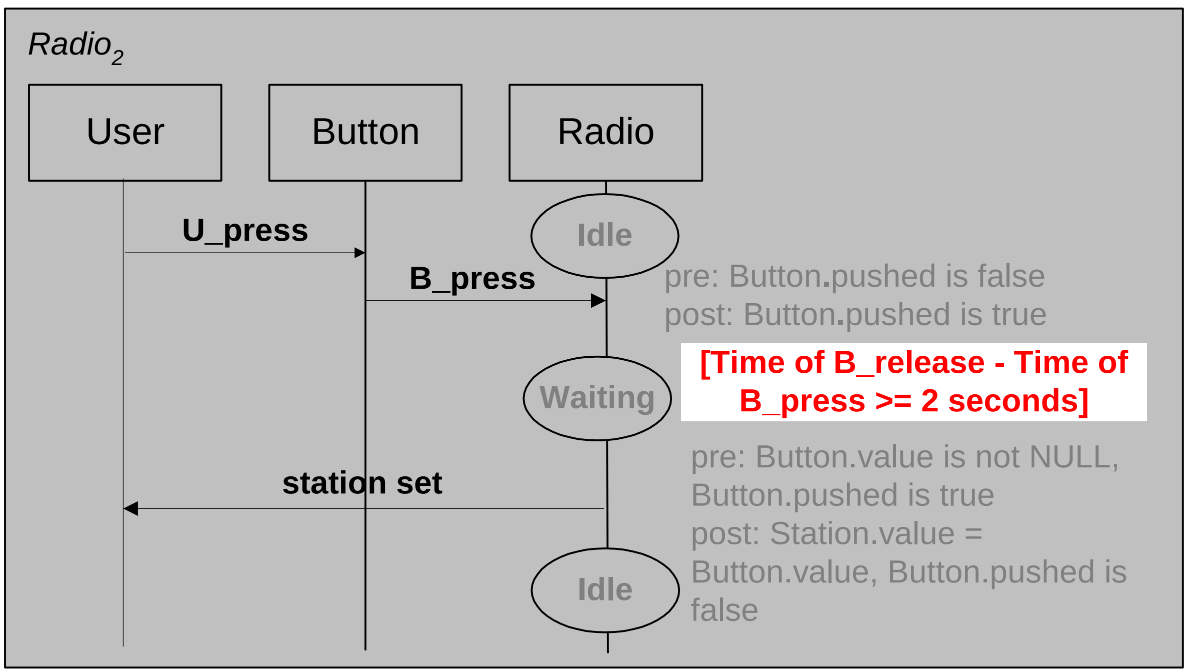




Too Much Information?

- How can we minimize the work investment?
- Goal: Annotate minimal information required for statechart synthesis
- May be additional goals that mandate more detail





Not if We Use Formal Grammar!

- Identify locations to add information
- Verify that added information is sufficient
  - For this example, only timing information was needed.

Grammar Parsing 101 (a flashback to your past…)

- Tokens and rules
  - Token – meaningful unit
  - Rule – determines legal strings of token symbols

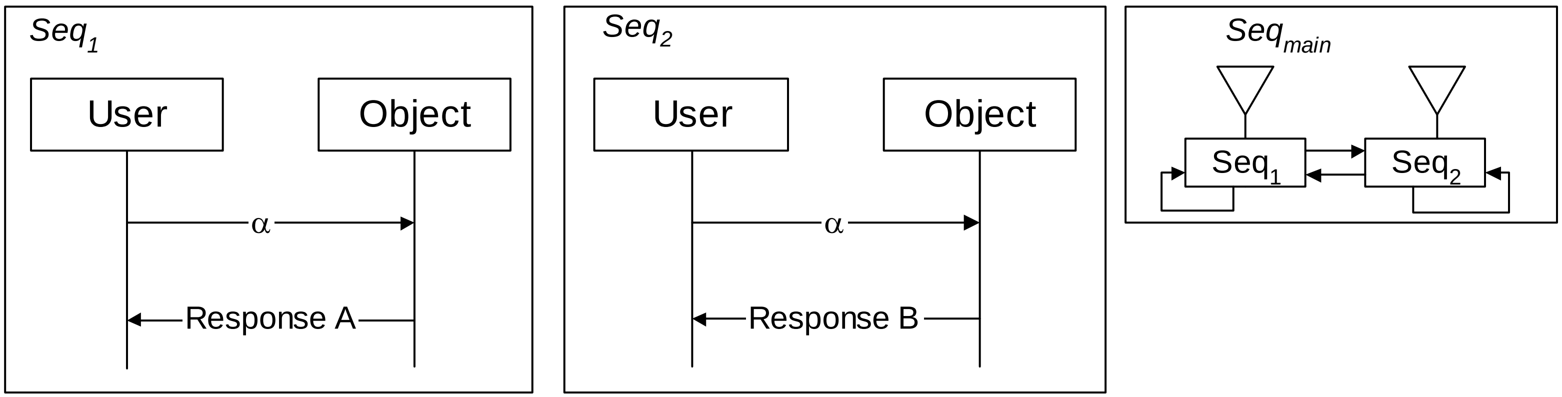
    SD $\rightarrow$ message **response** SD | ε
    message **response** $\rightarrow$ α **ResponseA** | α **ResponseB**

- Deterministic grammars
  - LL(1) – only one token needed to predict next step (deterministic)
  - LL(n) – need n lookahead (or backtrack)
    - Left-factoring - factor out shared terms
    - Backtracking - select a response, backtrack if incorrect

Formal Grammar Solution

- Token definitions:
  - Message set – consecutive information supplied to an object (eg, other objects' messages, time, state)
  - Response set – consecutive information generated by an object (eg, outgoing messages)
- Use grammar parsing to locate specification omissions
  - Omissions often result in non-determinism
  - Goal : one unique response set per unique message set
  - In formal terms, LL(1), if a message set is considered to be one item (otherwise LL(n) where n must be finite)

Generic Solution - Sequence Diagram Grammar



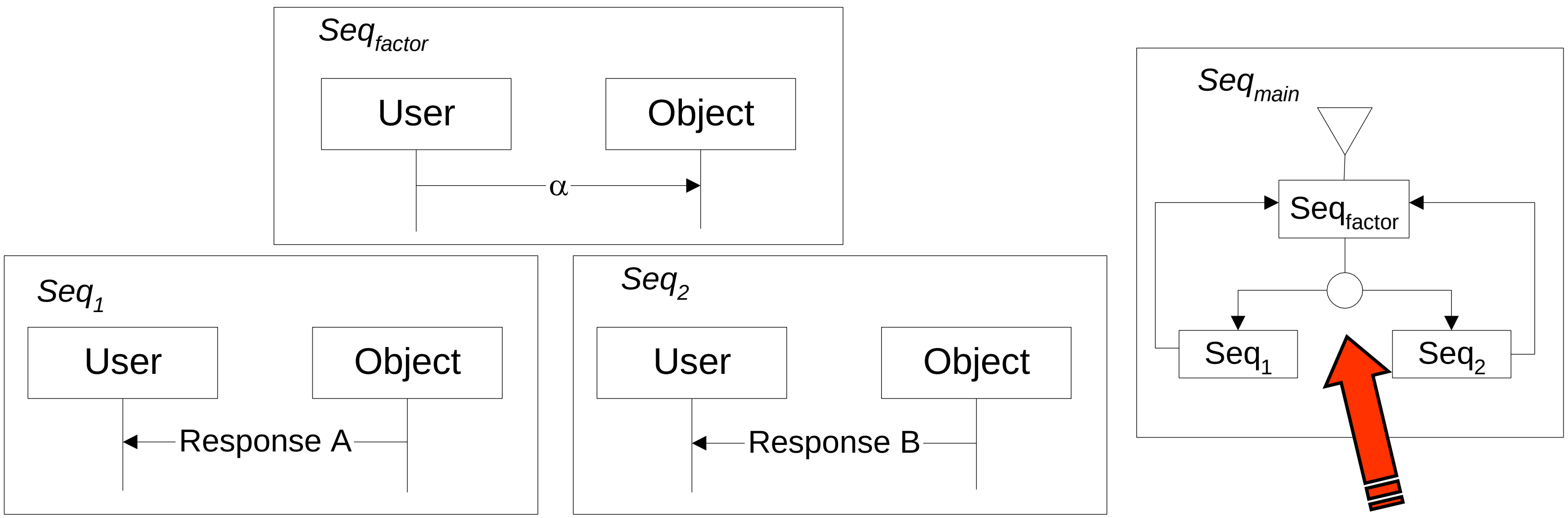
SD $\rightarrow$ message **response** SD | ε

message **response** $\rightarrow$ α **ResponseA** | α **ResponseB**

- The grammar highlights non-determinism here
  - Non-determinism is result of missing information, not grammar format
  - Left factoring, backtracking ineffectual

Why We Can’t Left Factor

- Left factoring *moves* non-determinism, doesn’t *remove* it



The non-determinism is now here

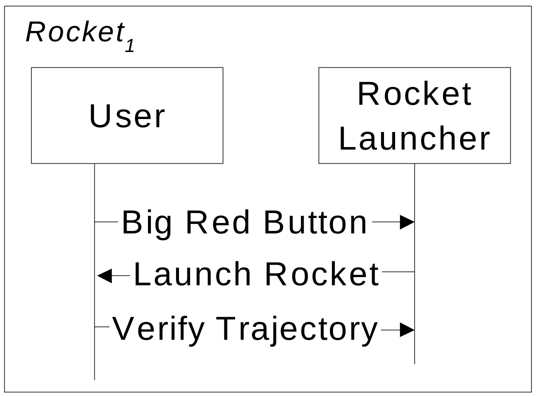
SD → message **response** SD | ε

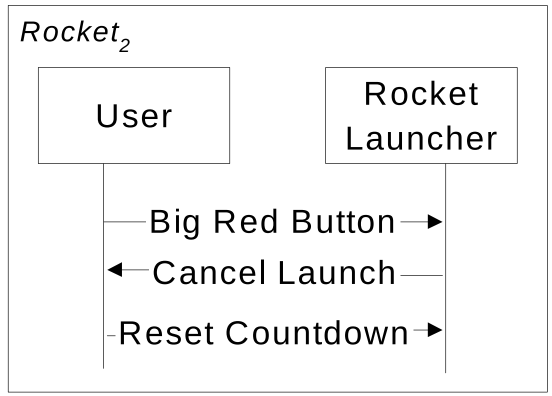
message **response** → α **A’**

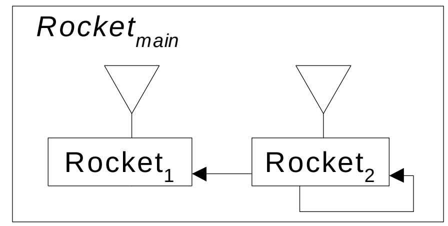
A’ → ResponseA | ResponseB

Why We Can't Backtrack

- Responses can't always be undone







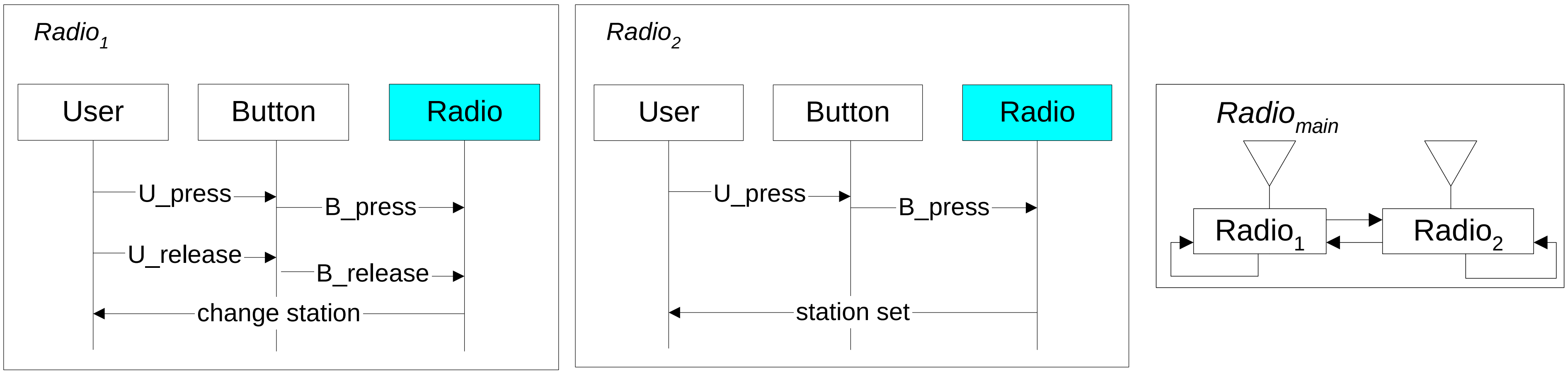
SD → message **response** SD | ε

message **response** → BigRedButton **LaunchRocket** VerifyTrajectory ε

| BigRedButton **CancelLaunch** ResetCountdown ε

- Also, possible to have only one message type

The non-deterministic car radio example...

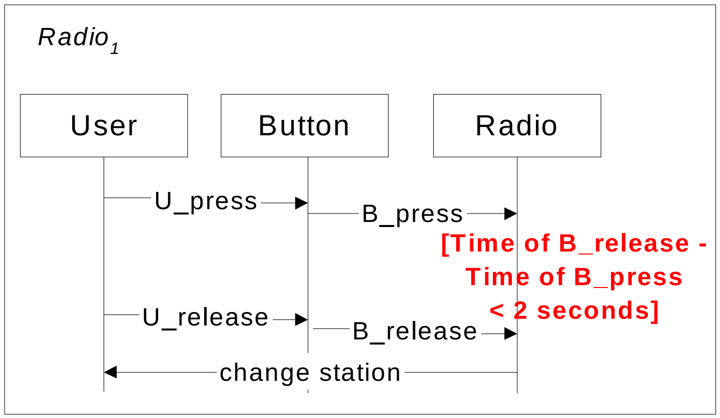


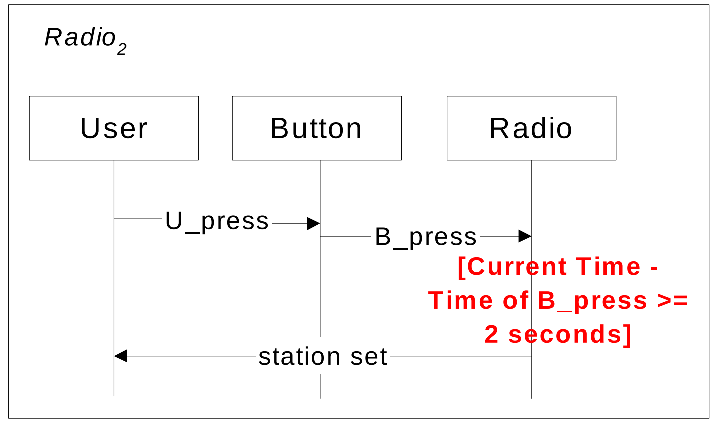
SD → message **response** SD | ε

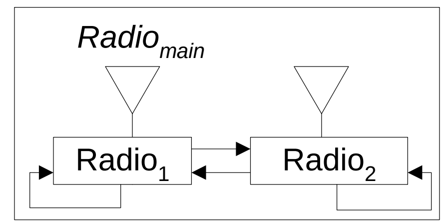
message **response** → B_press B_release **change_station**
| B_press **station_set**

message **response** → α B_release **change_station**
| α **station_set**

... becomes deterministic with timing information.







SD $\rightarrow$ message *duration* **response** SD | ε

message *duration* **response** $\rightarrow$

B_press *(Time of B_release – Time of B_press < 2 seconds)*
B_release **change_station**
| B_press *(Current Time – Time of B_press >= 2 seconds)*
station_set

message **response** $\rightarrow$ α B_release **change_station**
| β **station_set**

Additional Examples (in paper)

- Embedded examples
  - TV, power (state)
  - Elevator, floor (data)
- Automated Teller Machine (ATM) system
  - Apply technique to traditional transaction processing system example
  - Conclusions: Almost all unique message sets produced a unique set of system responses
    - Almost LL(1) already!
    - Notable exception: First response, Display main screen, followed the empty message set ε; only one initial condition so this is OK

Conclusions (1)

- In statechart synthesis from sequence diagrams, missing information may lead to unwanted non-determinism
  - Characteristics that exacerbate this:
    - Multiple initial conditions
    - Same user action evokes different response
    - Timing dependency
  - Common categories of additional information:
    - State, data, time

Conclusions (2)

- A formal grammar for sequence diagrams can locate non-determinism
  - Satisfies goals:
    - Minimal information annotation
    - Consistent screening method that can verify removal
  - Examples:
    - Car radio – representation and analysis
    - Can’t use left factoring, backtracking to eliminate non-determinism – need additional information!