鄭研究室

MTSA講座

～MTSAで制御器を作れるようになろう～

山内 拓人

離散事象システム

❖ 離散事象システム[1]

・事象の発生により状態が離散的に遷移するシステム

・状態遷移モデルで表現可能

・多くのソフトウェアシステムが該当

例）通信システム，業務システム，交通システム

❖ 安全性[2]が保障された離散事象システム

・制御不可な事象がどのように発生したとしても，**悪い事象が発生しない**システム

> 安全性 (safety) : 「悪い事象が発生しない」

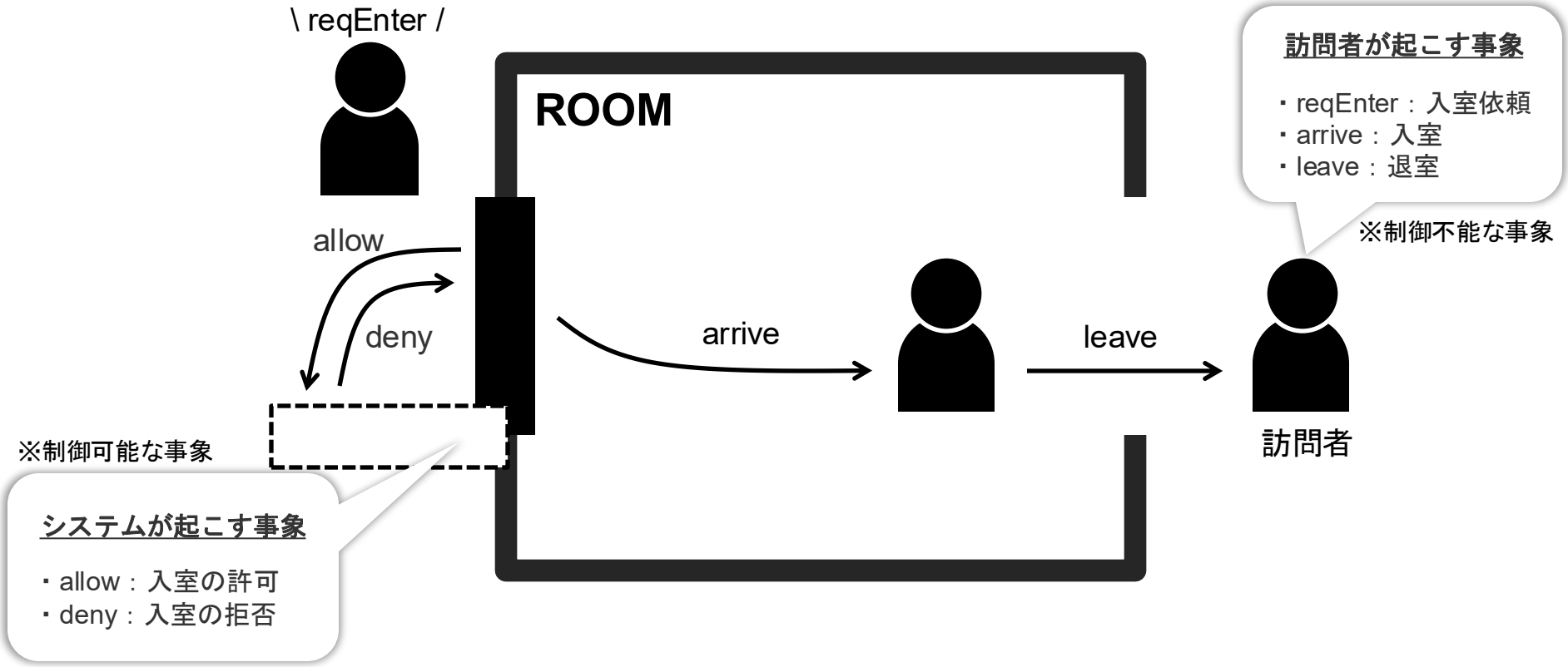
> 違反状態：悪い事象が生じた後の状態

[1] Cassandras, C. G.; Lafortune, S. Introduction to Discrete Event Systems. Springer US, 2008
[2] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. Distributed Computing, 2(3):117126, 1987.

離散事象システム

例 人数管理システム[1]

- 部屋への入室を許可/拒否することで人数管理
- 展示品の窃盗リスク，感染症の感染リスクの回避が目的

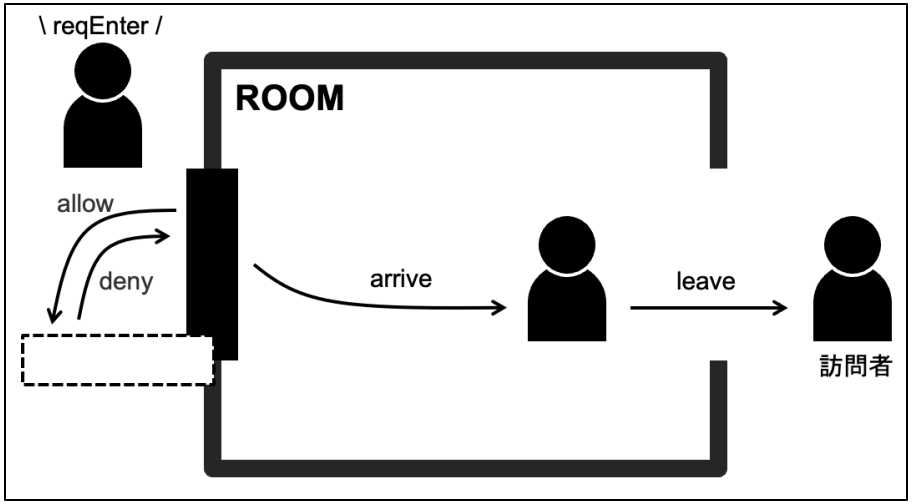


[1] A. Borda, et. al., Compositional verification of self-adaptive cyber-physical systems. IEEE/ACM 13th SEAMS, 2018

離散事象システムの開発
> 環境モデルの設計

❖ 環境を**環境モデル**としてモデル化

人数管理システム



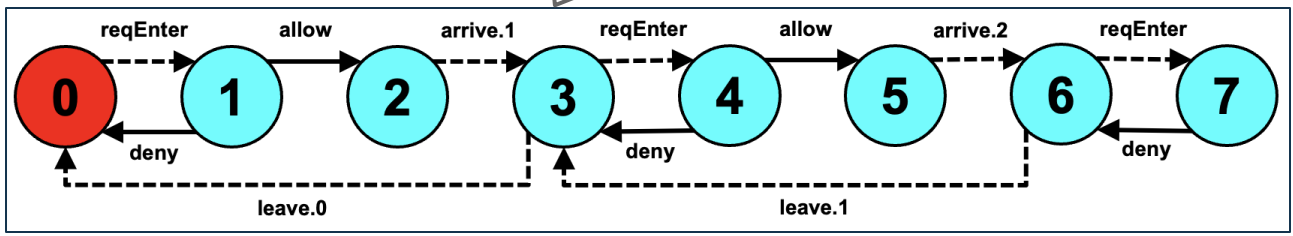
LTS：Labeled Transition System [1]

$$m = (S, A, \Delta, s_0)$$

S：状態
A：事象（$A = A^+ \cup A^-$）
A^+：制御可能な事象（実線矢印）
A^-：制御不能な事象（点線矢印）
Δ ： 遷移（$S \times A \times S$）
s_0：初期状態（$s_0 \in S$）

モデル化

ROOM：環境モデル



[1] Jeff Magee and Jeff Kramer. Concurrency: State Models and Java Programs. Wiley Publishing, 2006.

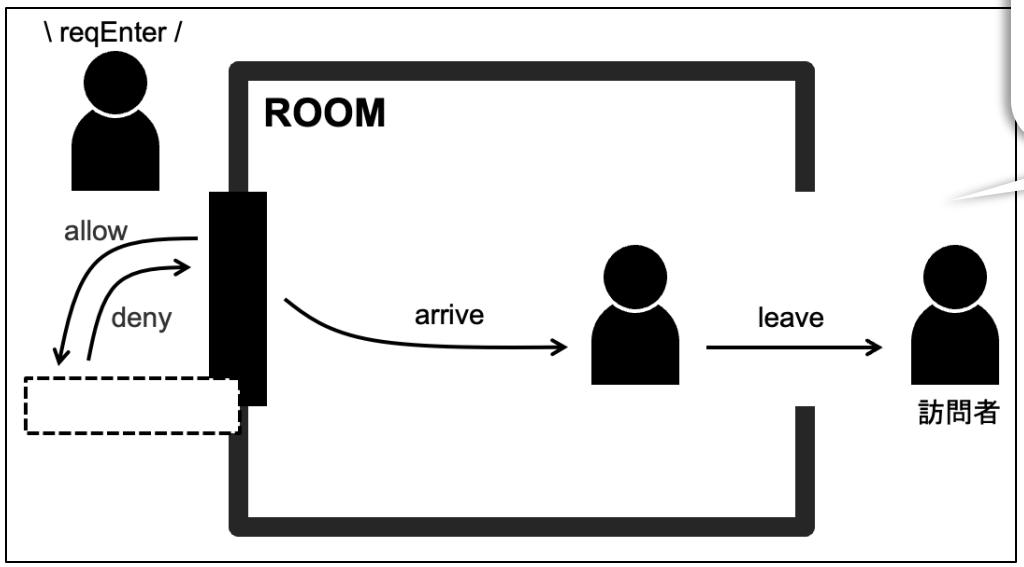
離散事象システムの開発
> 環境上で保障すべき安全性の設計

❖ 安全性[1]を**監視モデル**としてモデル化

監視モデル：悪い事象が発生すると違反状態に遷移する

違反状態：安全性を違反する状態（-1の状態）

人数管理システム

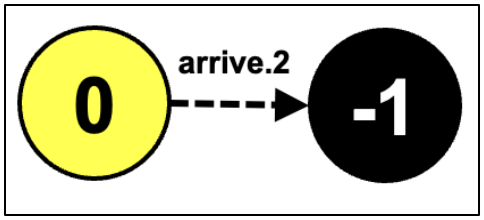


安全性

常に2人目を到着させない

モデル化

SAFETY：監視モデル



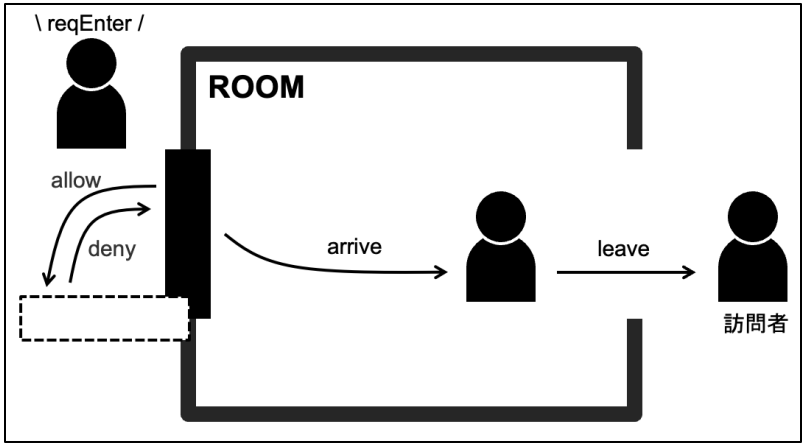
[1] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. Distributed Computing, 2(3):117126, 1987.

離散事象システムの開発
> システムの動作仕様の策定

❖ 安全性を考慮して**動作仕様**を手動で策定

動作仕様：システム動作に関わるソフトウェアの設計書

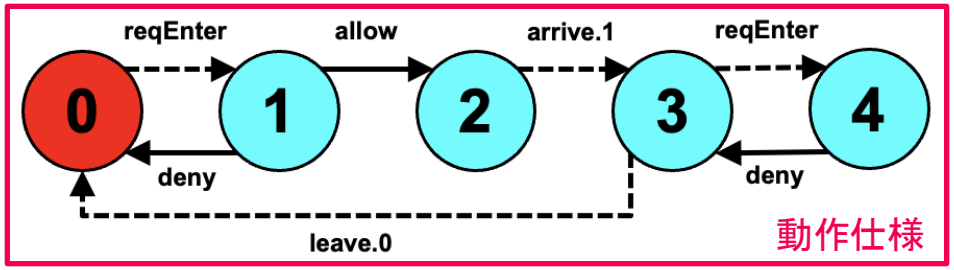
人数管理システム



安全性

常に2人目を到着させない

安全性を充足するよう
動作仕様を策定

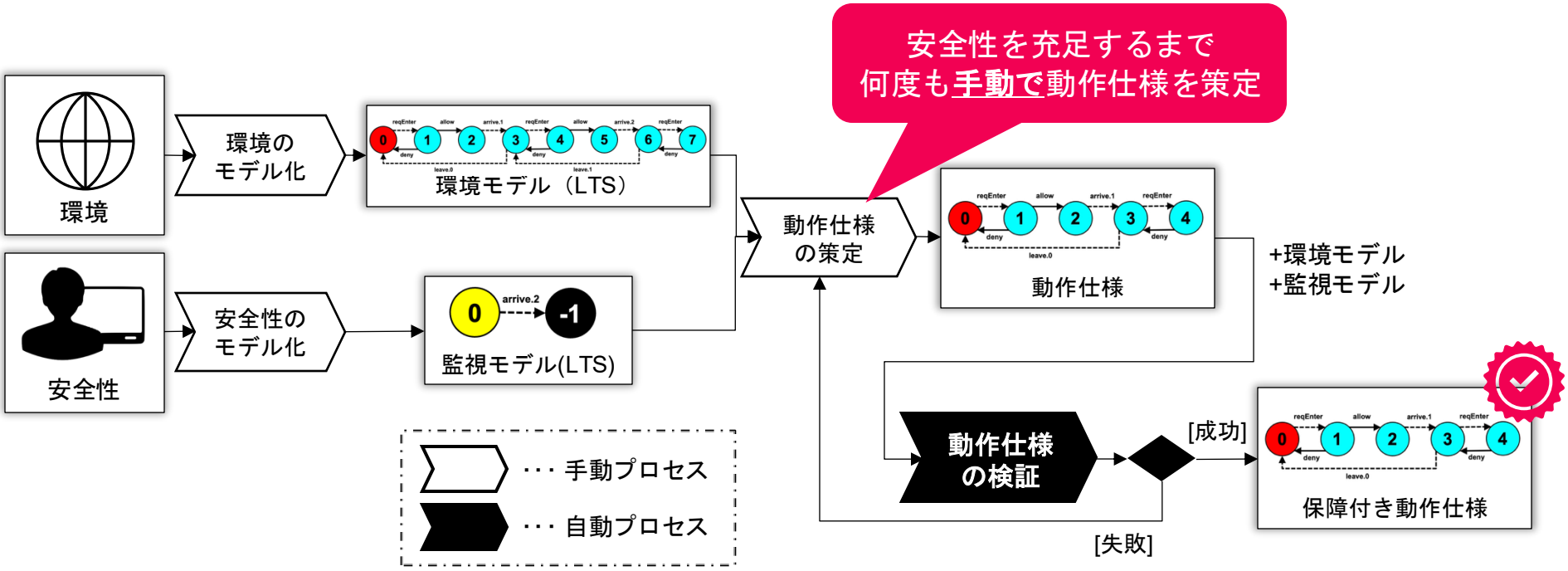


離散事象システムの開発
> モデル検査を用いた動作仕様の検証

❖ 動作仕様の検証（モデル検査）

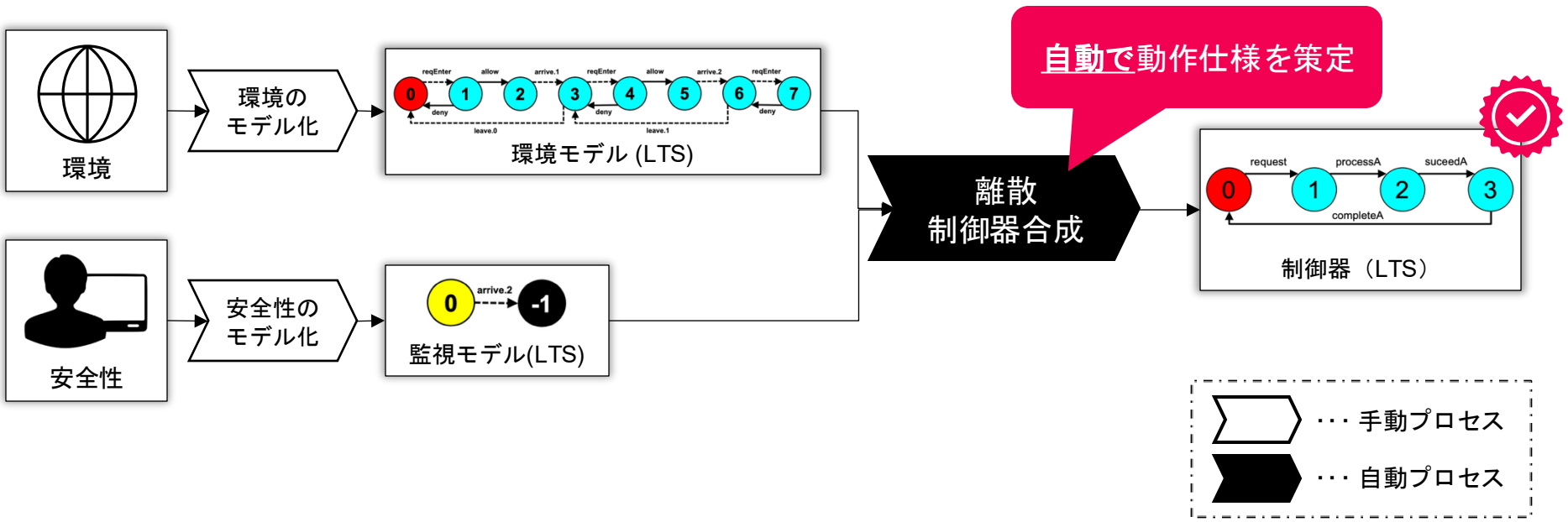
離散事象システムの動作仕様を検証

状態遷移モデルで表現された動作仕様を網羅的に検証する



離散制御器合成[1]
(DCS：Discrete Controller Synthesis)

- 環境モデルと監視モデルを与えることで,
  環境下で安全性が保証された制御器を自動で合成する.
  環境モデル：システムの動作環境をモデル化したLTS
  監視モデル：システムにおいて保証したい安全性をモデル化したLTS
  制御器：安全性の充足が保証された動作仕様（LTS）

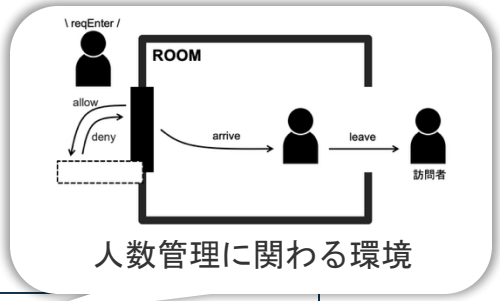


演習

STEP1:
環境モデルを作ってみよう！

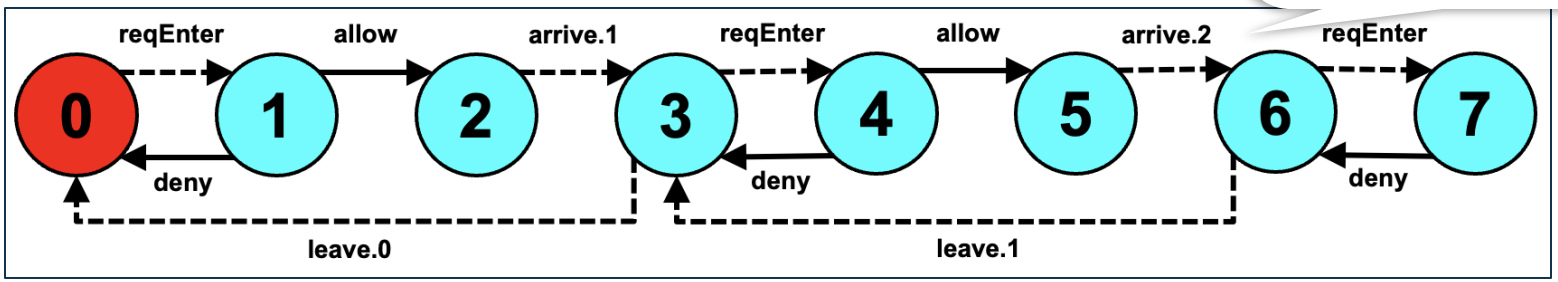
❖ 環境モデル

- 環境モデルとは「システムの**動作環境を表すLTS**」
- システムの構成要素（機能）ごとに環境モデルを設計する
- 安全性を充足している必要はない

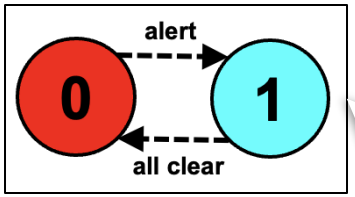


人数管理に関わる環境

ROOM：環境モデル



ALERT：環境モデル



非常時検知に関わる環境

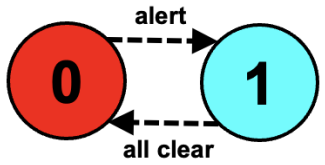
例 人数管理システムの環境モデル

STEP1:
環境モデルを作ってみよう！

❖ ALERTモデルの正解

```
1 /*環境モデル*/
2
3 ALERT = STATE0,
4 STATE0 = (alert -> STATE1),
5 STATE1 = (all_clear -> STATE0).
6
7 ||ALERT_MODEL= (ALERT).
```

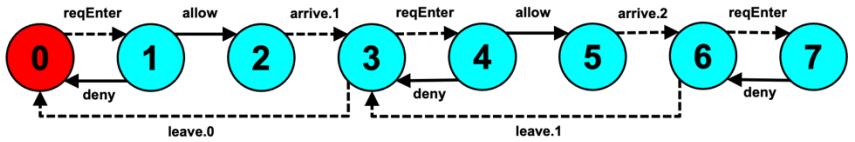
ALERT：環境モデル



STEP1:
環境モデルを作ってみよう！

❖ ROOMモデルの正解

ROOM：環境モデル




```
1  //解１
2  ROOM = STATE0,
3  STATE0 = (reqEnter -> STATE1),
4  STATE1 = (allow -> STATE2 | deny -> STATE0),
5  STATE2 = (arrive[1] -> STATE3),
6  STATE3 = (reqEnter -> STATE4 |leave[0] -> STATE0),
7  STATE4 = (allow -> STATE5 | deny -> STATE3),
8  STATE5 = (arrive[2] -> STATE6),
9  STATE6 = (reqEnter -> STATE7 |leave[1] -> STATE3),
10 STATE7 = (deny -> STATE6).
11
12
13 //解２
14 const X = 3 //変数
15 range N = 0..X //変数の範囲の宣言 ... 有限数の状態しか扱えない
16
17 ROOM = STATE0[0],
18 STATE0[n:N] = (reqEnter -> STATE1[n]
19                |when(n > 0) leave[n-1] -> STATE0[n-1]),
20 STATE1[n:N] = (when(n < X) allow -> arrive[n+1] -> STATE0[n+1]
21                |deny -> STATE0[n]).
22
23 ||ROOM_MODEL= (ROOM).
```

STEP2:
監視モデルを作ってみよう！

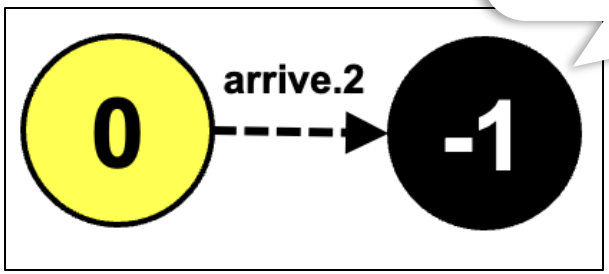
❖ 監視モデル

- 監視モデルシステムの**安全性を表すLTS**
- 悪い事象が発生すると違反状態に遷移するモデル
- 違反状態：悪い事象が生じた後の状態（-1の状態）
- 形式的表現である線形時相論理式から自動生成可能[2]

常に，arrive.2をしない

[]!arrive[2]

SAFETY：監視モデル



例 人数管理システムの監視モデル

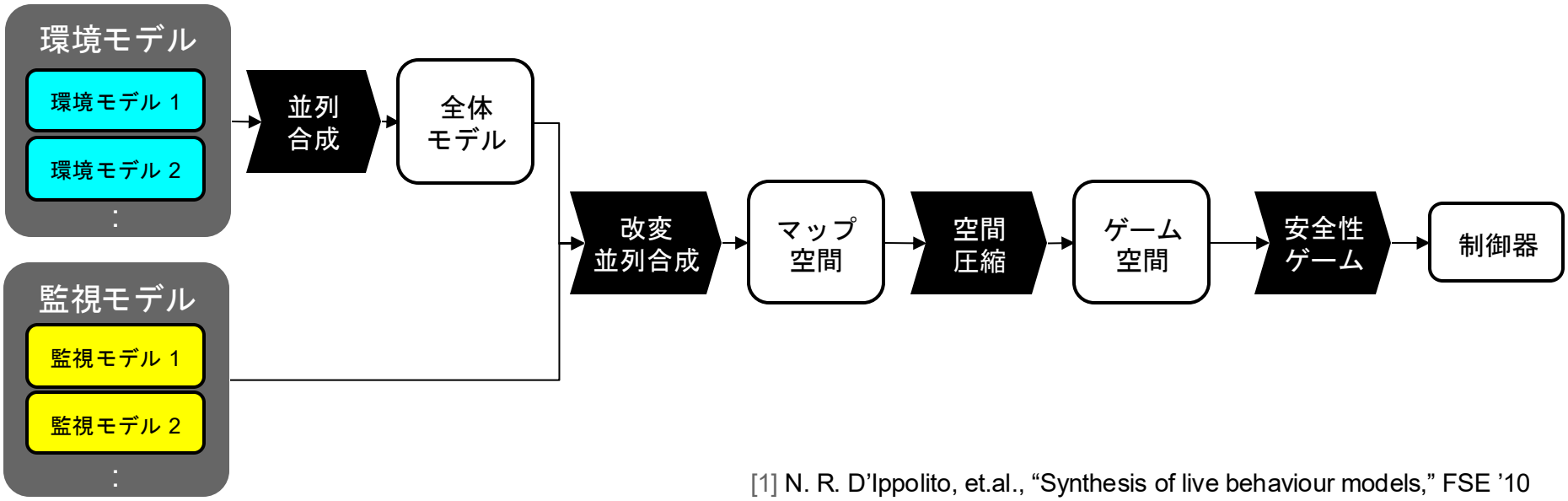
[1] N. R. D'Ippolito, et.al., "Synthesis of live behaviour models," FSE '10
[2] Yehia Abd Alrahman, et.al., Synthesis of run-to-completion controllers for discrete event systems.American Control Conference 2021.

離散制御器合成[1]の概要

❖ 離散制御器合成は４ステップで構成される

制御器を合成する上で３つの中間生成物が生成される

- 全体モデル：環境全体の振る舞いを表すLTS
- マップ空間：全体モデルに違反状態をマッピングしたLTS
- ゲーム空間：安全性ゲームを解くためのLTS



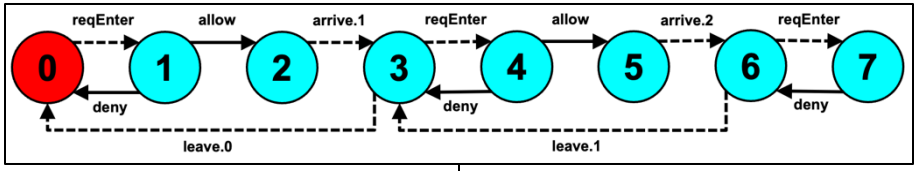
[1] N. R. D'Ippolito, et.al., "Synthesis of live behaviour models," FSE '10

離散制御器合成の要素技術
> 全体モデルの構築（Step1）

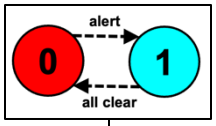
❖ 複数の環境モデルから**全体モデル**を構築

並列合成[1]：複数の環境モデルを一つに重ね合わせたモデルを構築

ROOM：環境モデル

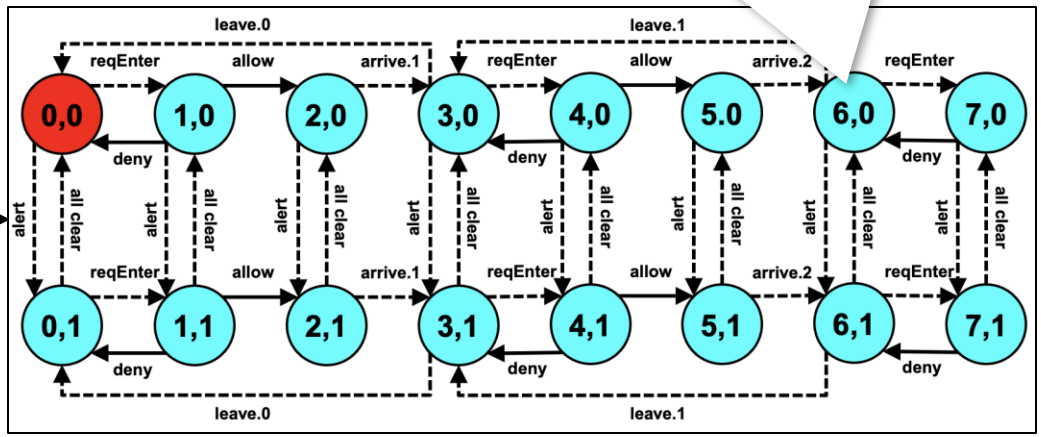


ALERT：環境モデル



並列合成

全体モデル



左の数字「6」：**ROOM**モデルの状態
右の数字「0」：**ALERT**モデルの状態

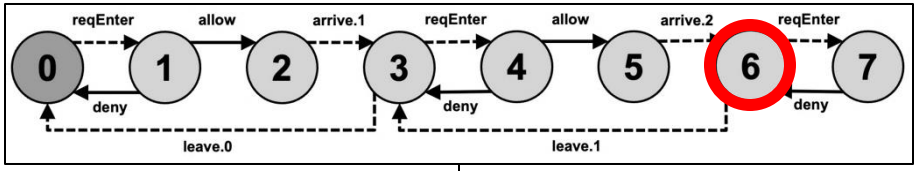
[1] Giannakopoulou, D. and Magee, J.: Fluent Model Checking for Event-based Systems, SIGSOFT Softw. Eng.

離散制御器合成の要素技術
> 全体モデルの構築（Step1）

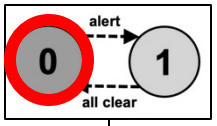
❖ 複数の環境モデルから**全体モデル**を構築

並列合成[1]：複数の環境モデルを一つに重ね合わせたモデルを構築

ROOM：環境モデル

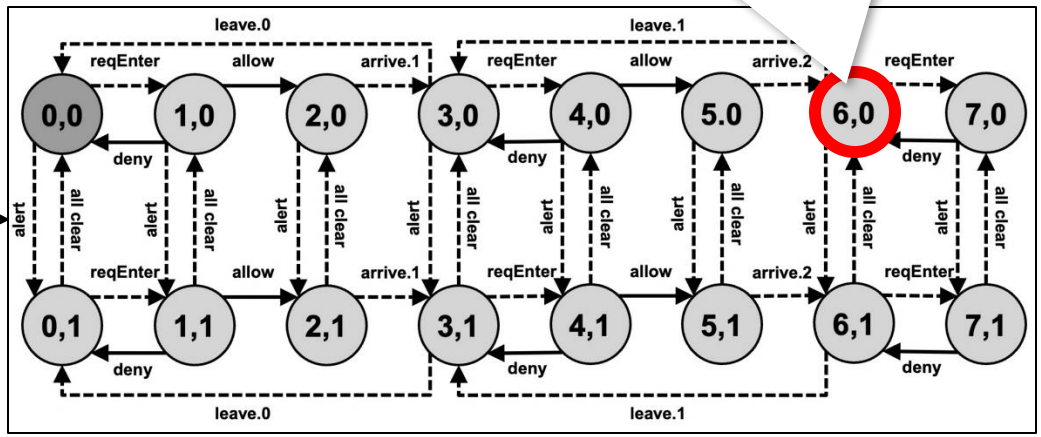


ALERT：環境モデル



並列合成

全体モデル



左の数字「6」：**ROOM**モデルの状態
右の数字「0」：**ALERT**モデルの状態

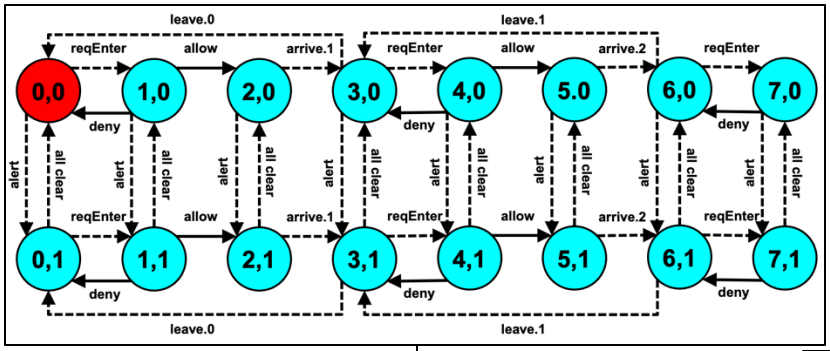
[1] Giannakopoulou, D. and Magee, J.: Fluent Model Checking for Event-based Systems, SIGSOFT Softw. Eng.

離散制御器合成の要素技術
> マップ空間の構築 （Step2）

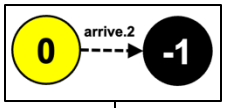
❖ 全体モデルと監視モデルから**マップ空間**を構築

改変並列合成[1]：全体モデルに監視モデルの状態を重ね合わせる

全体モデル

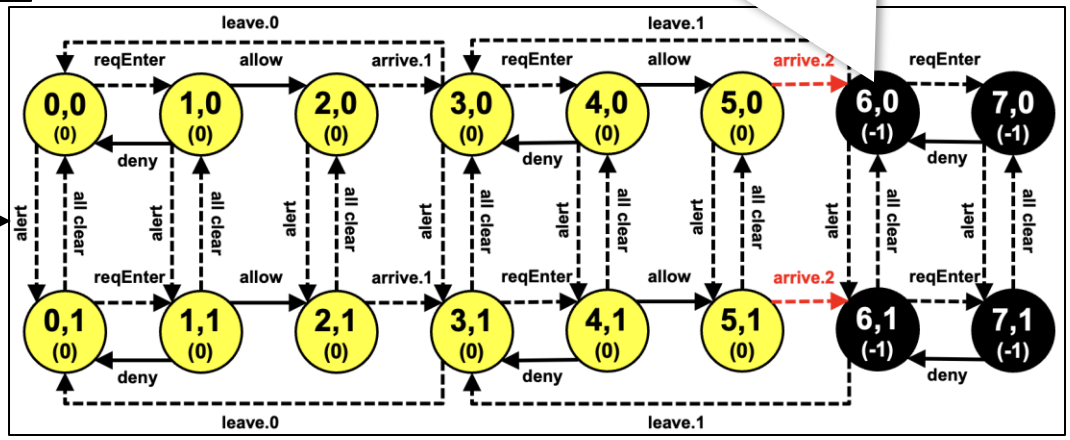


SAFETY：監視モデル



改変
並列合成

マップ空間



左の数字「2」：**ROOMモデルの状態**
右の数字「0」：**ALERTモデルの状態**
下の数字「0」：**SAFETYモデルの状態**

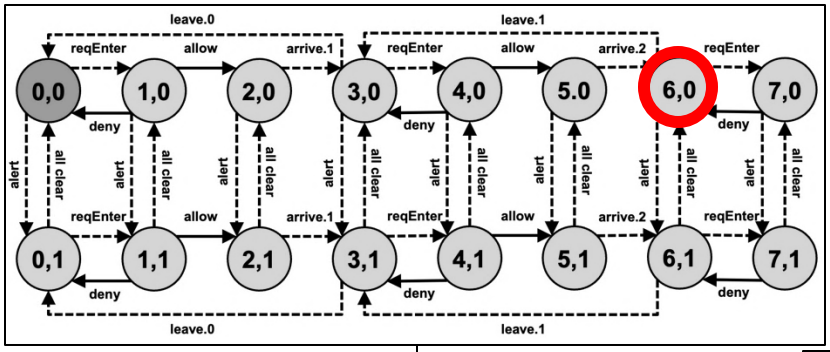
[1] Nicol´as Roque D'Ippolito, Victor Braberman, Nir Piterman, and Sebasti´an Uchitel. Synthesis of live behaviour models., FSE2010.

離散制御器合成の要素技術
> マップ空間の構築 （Step2）

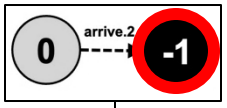
❖ 全体モデルと監視モデルから**マップ空間**を構築

改変並列合成[1]：全体モデルに監視モデルの状態を重ね合わせる

全体モデル

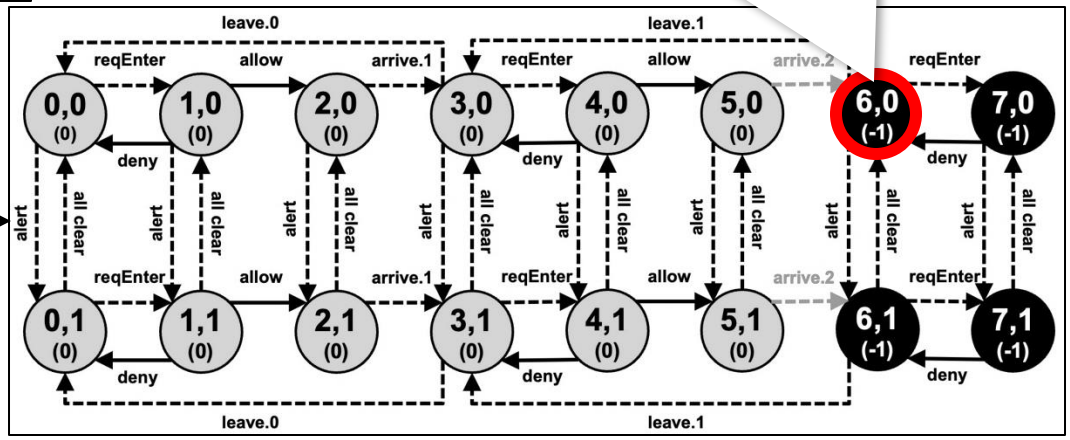


SAFETY：監視モデル



改変
並列合成

マップ空間



左の数字「2」：**ROOM**モデルの状態
右の数字「0」：**ALERT**モデルの状態
下の数字「-1」：**SAFETY**モデルの状態

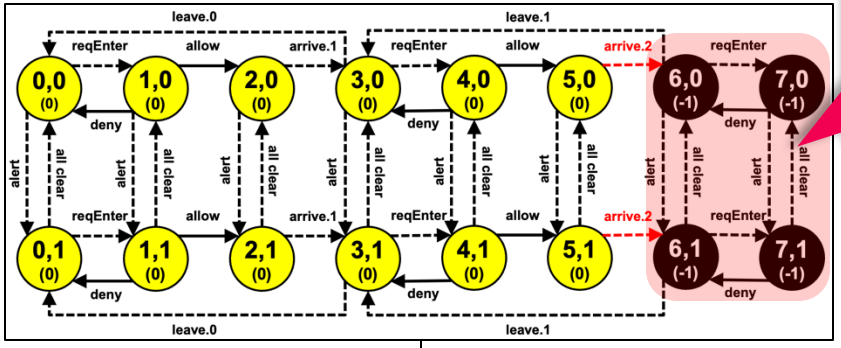
[1] Nicol´as Roque D'Ippolito, Victor Braberman, Nir Piterman, and Sebasti´an Uchitel. Synthesis of live behaviour models., FSE2010.

離散制御器合成の要素技術
> ゲーム空間の構築 （Step3）

❖ マップモデルから**ゲーム空間**を構築

空間圧縮：マップ空間のうち安全性を違反する状態（-1を含む状態）をひとつの状態に圧縮

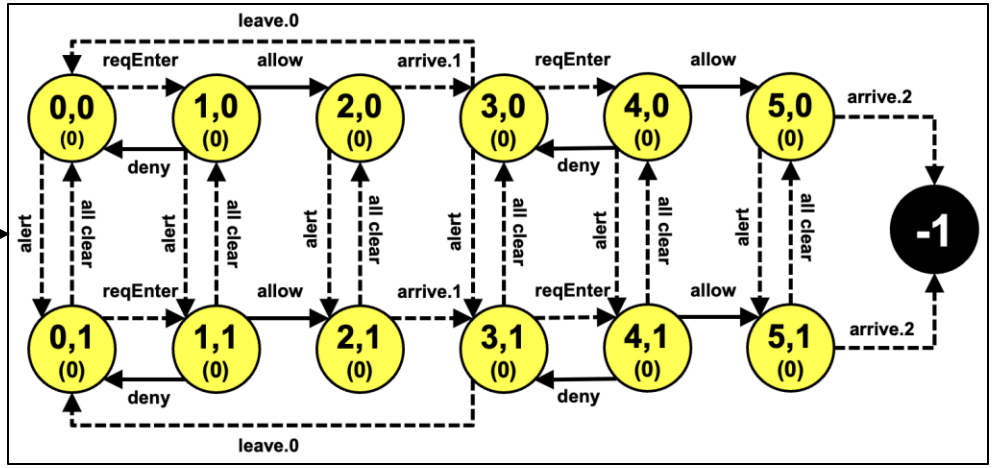
マップ空間



監視モデルにおいて「-1」となる状態
（黒く塗りつぶされた状態）をひとつの状態に圧縮する

空間圧縮

ゲーム空間



[1] Nicol´as Roque D'Ippolito, Victor Braberman, Nir Piterman, and Sebasti´an Uchitel. Synthesis of live behaviour models., FSE2010.

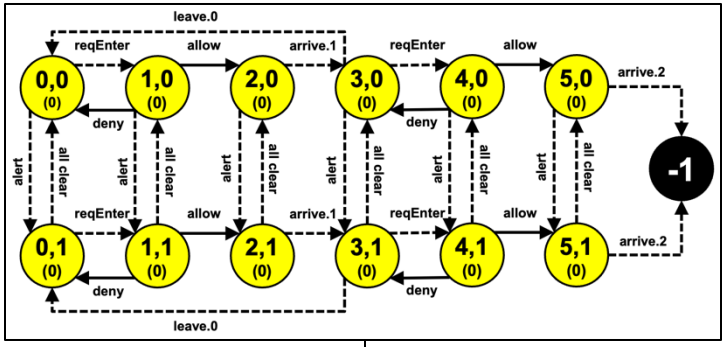
離散制御器合成の要素技術

> 制御器の導出 （Step4）

❖ ゲーム空間から**制御器**を導出

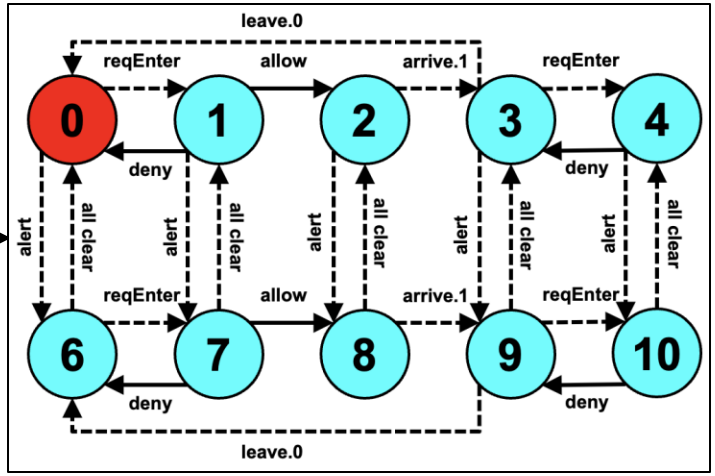
安全性ゲーム[1]：二人型対戦ゲーム理論に基づき，制御不可能な事象がどのように生じたとしても，違反状態に到達しない状態と遷移を導出

ゲーム空間



安全性ゲーム

制御器



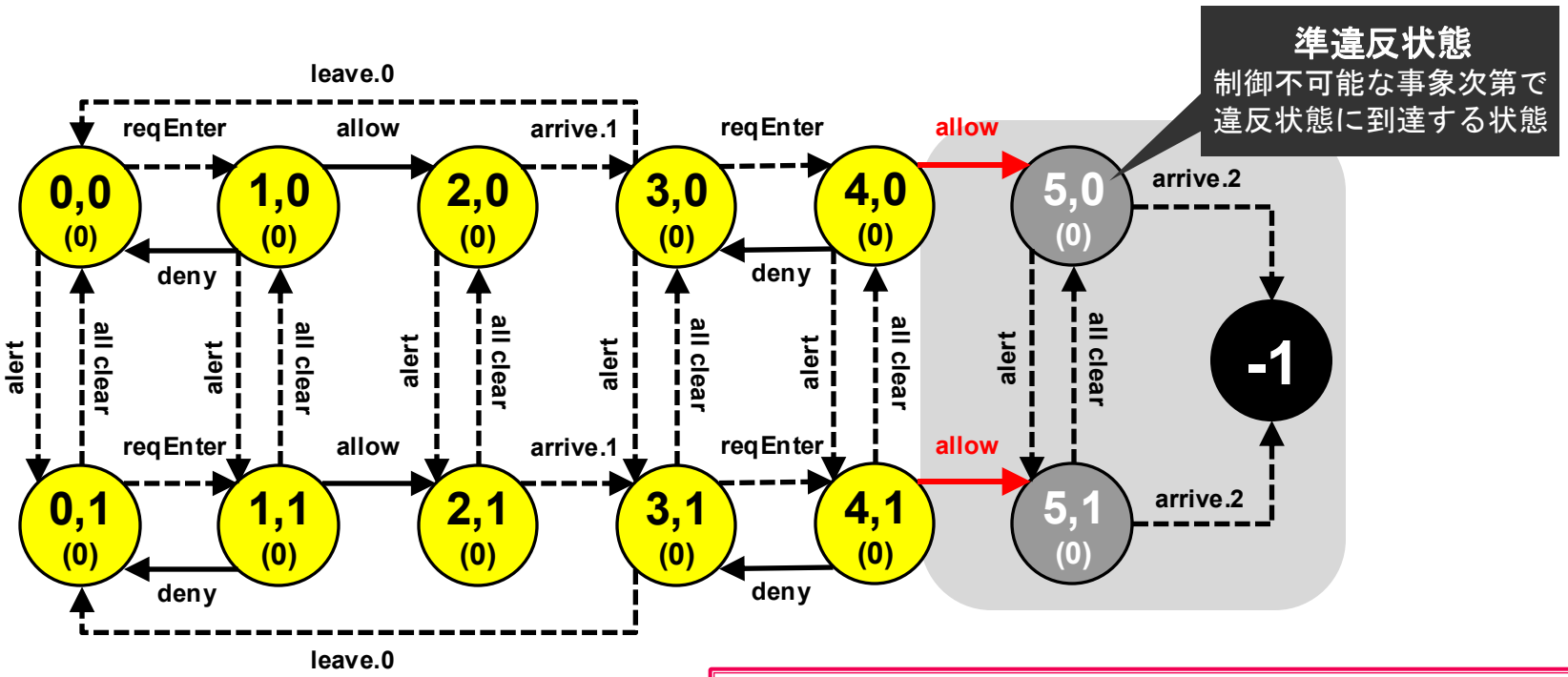
[1] Nicol´as Roque D'Ippolito, Victor Braberman, Nir Piterman, and Sebasti´an Uchitel. Synthesis of live behaviour models., FSE2010.

離散制御器合成の要素技術

> 制御器の導出 （Step4）

❖ ゲーム空間から**制御器**を導出

安全性ゲーム[1]：二人型対戦ゲーム理論に基づき，制御不可能な事象がどのように生じたとしても，違反状態に到達しない状態を導出する.



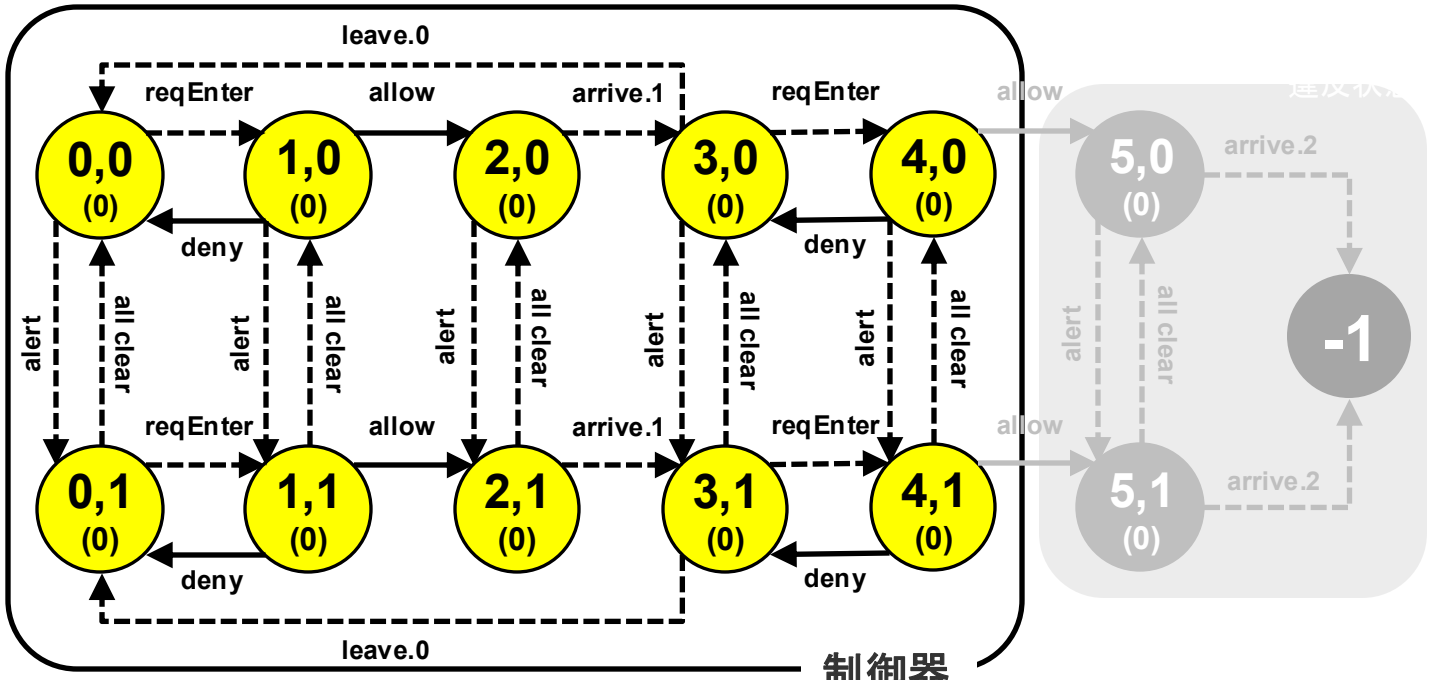
状態(4,0)と状態(4,1) で**allow**をしなければ，
違反状態に到達することはない

離散制御器合成の要素技術

> 制御器の導出 （Step4）

❖ ゲーム空間から**制御器**を導出

安全性ゲーム：二人型対戦ゲーム理論に基づき，制御不可能な事象がどのように生じたとしても，違反状態に到達しない状態を導出する.



＝違反状態に到達することのない
環境全体の振る舞い

離散制御器合成における
計算空間

❖ DCSにおいてボトルネックとなる状態空間

中間生成物である全体モデル, マップ空間, ゲーム空間が該当

→入力の環境モデルの数に対して**指数増加する課題**あり