離散事象システムに基づいた人間と調和したマルチロボットによる倉庫自動化に関する研究

令和２年度

大阪市立大学 工学部 電気情報工学科
エレクトロニクス領域

野田 健太朗

概 要

物流業界の人手不足とさらに 2020 年に流行した感染症の影響で世界中でネットショッピングの利用率の増加 [1] に伴って，無人搬送ロボットを導入した倉庫自動化による作業の効率化が進んでいる．

しかし，すべてロボットだけで完結する完全な倉庫の自動化の実現は困難である．無人搬送ロボットは荷物をピッキングして運搬する機能はあっても，予期せぬトラブルが発生した場合，それを対処することが難しい．また，ロボットには，さまざまな大きさや形のもののピッキング，及び想定されていなかったタスクを処理することが困難である．

ここで，近年注目されている，自動化された一環のシステムに人間を介入させたシステムを考える Human-in-the-Loop という概念を取り入れることで，手先の器用さ，問題解決能力などの人間の強みと，正確性，パワーやスピードなどのロボットの強みをかけ合わせることにより予期せぬ事態に対応できると考えた．従来では倉庫内のトラブルを解決処理しに人間が立ち入るとき，自動化されたシステムを停止していたが，人間をシステムの一部と考えることでロボットの動作中にトラブルを処理できるようにする．

このとき人間の安全を確保したシステムをつくる必要があり，その制御方法について，人間の経路のロボットの使用を禁止する方法と，人間にロボットを接近させない方法を提案する．また，ケーススタデによって検証し，2 つの方法が有効であり，人間の安全を確保できることを示した．

目次

図目次

表目次

第 1 章

序論

1.1　背景

近年，実店舗に出向かなくても買い物ができるネットショッピングが増加している．

さらに，2020 年に入ってから COVID-19 の影響もあり，さらに拍車をかけるようにネットショッピングが買い物の主流な形になるようになった．それにより，物流業界では人手不足が深刻化している．それを解決するのが移動式ロボットによる倉庫の自動化である．それはつまり，人が目的の荷物を探すため歩き回る必要があったが，その仕事をロボットが代わりに行うということである．このように倉庫内で無人で荷物を運搬するロボットを活用することが，労働の削減につながり，さらには労働者不足を解消することにつながる．

倉庫内のロボットを制御するうえで，ロボット同士の衝突，ロボットで道がふさがれていまいお互い身動きができなくなるデッドロック状態などの問題が考えられる．それらの問題を解決する有効な方法として，離散事象システムのスーパーバイザ制御が挙げられる [2][3].

先行研究に，スーパーバイザ制御の計算に特化した TCT という計算ソフトウェアを使用して，自動化倉庫のスーパーバイザ制御を設計する研究がある [4]．これは，ロボットごとの動的オートマトンモデルを作り，衝突やブロッキング状態が発生してしまうロボッ

トの動きを制限して，タスクを各ロボットが円滑にこなせるように，スーパーバイザを計算するアルゴリズムが提案されている．

倉庫自動化の実用化において，ロボットで対応できないトラブルが発生したとき，どう対処するかが課題となっている．現在，そのようなトラブルが発生して，処理するために人が立ち入るとなると，自動化されたシステムを完全に停止させなければならない．これは効率化を目指すうえで解決しなければならない問題である．

1.2 目的

本研究では，システムを停止せずに，人間が倉庫に立ち入ることを想定したケースにおいても，離散事象システムを用いたスーパーバイザ制御を応用できることを追究する．倉庫内で発生した予測不可能なトラブルを人間が処理するとき，ロボットも使用する通路を通りトラブルの起こった場所まで行かなければならない．つまり，人間も考慮したうえでロボットの制御を行わなくてはならない．このように，自動化された一環のシステムに，人間を介入させてシステムを構築する Human-in-the-loop という近年注目を浴びており，センシングや機械学習などさまざまな分野で応用されている [5][6][7][8]．本研究では倉庫の自動化にこの考えを取り入れた．

人間がロボットの動き回る倉庫内に入る制御を考えるとき，最も重要なのは人間の絶対的な安全を確保しなければならないということである．ロボットが通常通りせかせかと運搬している倉庫内に人間が立ち入るのは危険で，人間の安全な領域を確保しなければならない [9]．予期せぬトラブルや特殊な作業など，人間が行わなければならないタスクが発生したとき，自動化されたシステムを完全に停止することなく処理するには，人の安全を保障できるロボットの制御が求められる．そのため，人の安全を条件とした倉庫内の荷物運搬ロボットの制御を Human-in-the-loop の考えを取り入れたスーパーバイザ制御理論で考える．

論文の構成は次のようになっている．第 2 章は，スーパーバイザ制御理論の基礎を述べる．第 3 章は，倉庫の自動化についてモデリングし，スーパーバイザ制御理論を適応させる方法を紹介する．第 4 章は，人間を考慮したシステムを制御する方法を 2 つ提案して説明する．第 5 章では，シミュレーションを行い，システムの安全性について検証する．第

6 章は，まとめと今後の課題について述べる．

第 2 章

スーパーバイザ制御理論

この章では，離散事象システムとオートマトン，およびスーパーバイザ制御の計算方法を解説する．

2.1 オートマトン

2.1.1 有限オートマトンの基本

離散事象システムとは，離散的な状態集合を持つ事象駆動型システムである [10]．事象が生起すると，その事象に依存して状態が変わり，その状態が時間的に持続するのが特徴である．オペーションシステム，データベースシステム，通信システム，生産システム，シーケンス制御システムなどが離散事象システムとみなせるいい例である [11].

離散事象システムのモデリングには有限オートマトンが適している．有限オートマトンとは有限個の状態と事象が定義されている動的システムのことである．生起した事象と状態遷移関数に従って，現在いる状態を遷移し連続する一連の動作（ふるまい）をモデリングする．

有限オートマトン G は次のように 5 つの要素から表される．

$$G = (Q, \Sigma, \delta, q_0, Q_m) \tag{2.1}$$

ここで Q は状態の集合, Σ は事象の集合, δ は状態遷移関数, $q_0 \in Q$ は初期状態, $Q_m \subseteq Q$

は受理状態の集合を表している．状態遷移は $\delta(q,\sigma)$ と書き，さらに状態 q において事象 σ が生起しうることを $\delta(q,\sigma)$ が定義されているといい，以下のように表される．

$$\delta(q,\sigma)!$$

k 個の事象 $\sigma_i \in \Sigma(i=1,\cdots,k)$ において，$s=\sigma_1\cdots\sigma_k$ とならべたものを事象列 s と呼ぶ．また，事象列の事象の数を，事象列の長さといい，$\mid s \mid$ と書くことで s の長さを表す．

Σ に含まれる事象で作られるすべての事象列の集合を Σ^* とかく．この Σ^* には，長さ 0 の事象列である空事象列 ϵ も含まれている．

状態遷移関数 $\delta : Q\times\Sigma\to Q$ を $\hat{\delta} : Q\times\Sigma^*\to Q$ のように，以下の式の通りに拡張する．

$$\begin{aligned}\hat{\delta}(q,\epsilon) &= q\\ \hat{\delta}(q,\sigma) &= \delta(q,\sigma), q\in Q, \sigma\in\Sigma\\ \hat{\delta}(q,s\sigma) &= \delta(\hat{\delta}(q,s),\sigma), q\in Q,\sigma\in\Sigma, s\in\Sigma^*\end{aligned}$$

そして以後，$\hat{\delta}$ のハットを省略して δ とかく．

2.1.2 言語

Σ^* の任意の部分集合を事象集合 Σ の言語 L という．

$$L\subseteq\Sigma^* \tag{2.2}$$

ある集合を与えられたとき，その集合のすべての部分集合から構成される集合をべき集合という．そして Σ^* のべき集合を $Pwr(\Sigma^*)$ とかき，L は以下のようにも記すことができる．

$$L\in Pwr(\Sigma^*) \tag{2.3}$$

事象列 $s\in\Sigma^*$ を $s=s_1s_2$ とかくことができる事象列 $s_2\in\Sigma^*$ が存在するとき事象列 $s_1\in\Sigma^*$ を s の接頭語とよぶ．言語 L に含まれる事象列のすべての接頭語からなる言語 $\overline{L}$

を次のように表す.

$$\overline{L} = \{s_1 \in \Sigma^* \mid (\exists s_2 \in \Sigma^*) s_1 s_2 \in L\} \tag{2.4}$$

オートマトン G に対して，起こり得るすべての事象集合を $L(G)$ と定義する.

$$L(G) \coloneqq \{s \in \Sigma^* \mid \delta(q_0, s)!\} \subseteq \Sigma^* \tag{2.5}$$

また，$L(G)$ のうち，初期状態から受理状態まで遷移する事象列の集合を $L_m(G)$ といい以下のように定義する.

$$L_m(G) \coloneqq \{s \in L(G) \mid \delta(q_0, s) \in Q_m\} \subseteq L(G) \tag{2.6}$$

$\overline{L_m(G)} = L(G)$ が成り立つ場合にのみ， G はノンブロッキングだといえる.

2.2 オートマトンの計算

2.2.1 自己ループ

オートマトン $G = (Q, \Sigma, \delta, q_0, Q_m)$ をおいたとき，$\Sigma' \cap \Sigma = \emptyset$ である事象の集合 Σ' との自己ループ G_{sl} は次のようなオートマトンとなる.

$$G_{sl} = (Q, \Sigma \dot{\cup} \Sigma', \delta \dot{\cup} \delta', q_0, Q_m) \tag{2.7}$$

ただし，$\delta' \coloneqq \{[q, \sigma', q] \mid q \in Q, \sigma' \in \Sigma'\}$. つまり，自己ループというのは，もともと定義された G に関係のない事象 σ' が起こったとき，G は同じ状態に遷移するという操作（計算）のことである.

2.2.2 自然な射影

自然な射影 $P\colon (\Sigma \dot{\cup} \Sigma')^* \to \Sigma^*$ を以下のように定義する.

(i) $P(\epsilon) = \epsilon$

(ii) $\sigma \in \Sigma$ のとき $P(\sigma) = \sigma$
$\sigma \in \Sigma'$ のとき $P(\sigma) = \epsilon$

(iii) $s \in \Sigma^*, \sigma \in \Sigma \cup \Sigma'$ について $P(s\sigma) = P(s)P(\sigma)$

この $P(\cdot)$ は，事象列を与えられたとき，特定の事象集合に含まれている事象だけを時系列順に取り出す関数である．

また，言語 $L \in Pwr(\Sigma^*)$ について，自然な射影 P の逆関数 $P^{-1}\colon Pwr(\Sigma^*) \to Pwr((\Sigma \dot{\cup} \Sigma')^*)$ は以下のようになる．

$$P^{-1}(L) = \{s \in (\Sigma \dot{\cup} \Sigma')^* \mid P(s) \in L\} \tag{2.8}$$

2.2.3 同期合成

同期合成とは，複数のオートマトンから新しい 1 つのオートマトンを生み出す計算で，オートマトン G_1, G_2 の同期合成 G は次のように定義される．

$$G_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1}) \tag{2.9}$$
$$G_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2}) \tag{2.10}$$
$$G = G_1 \mid\mid G_2 = (Q, \Sigma, \delta, q_0, Q_m) \tag{2.11}$$

G_1 は $\Sigma_2 - \Sigma_1$ に対して，G_2 は $\Sigma_1 - \Sigma_2$ に対して自己ループオートマトン G_{sl1}，G_{sl2} を生成する．

$$G_{sl1} = (Q_1, \Sigma_1 \cup (\Sigma_2 - \Sigma_1), \delta_{sl1}, q_{0,1}, Q_{m,1}) \tag{2.12}$$
$$G_{sl2} = (Q_2, \Sigma_2 \cup (\Sigma_1 - \Sigma_2), \delta_{sl2}, q_{0,2}, Q_{m,2}) \tag{2.13}$$

ここで，δ_{sl1}，δ_{sl2} は以下のようである．

$$\delta_{sl1}(q, \sigma) = \begin{cases} \delta_1(q, \sigma) & (\sigma \in \Sigma_1) \\ q & (\sigma \notin \Sigma_1) \end{cases} \tag{2.14}$$

$$\delta_{sl2}(q, \sigma) = \begin{cases} \delta_1(q, \sigma) & (\sigma \in \Sigma_2) \\ q & (\sigma \notin \Sigma_2) \end{cases} \tag{2.15}$$

この 2 つをもとに次のように $G_1||G_2$ を計算する．

$$G = G_{sl1} \times G_{sl2} \tag{2.16}$$

$$L(G_1 \mid\mid G_2) = [P_1^{-1}L(G_1)] \cap [P_2^{-1}L(G_2)] \tag{2.17}$$

$$L_m(G_1 \mid\mid G_2) = [P_1^{-1}L_m(G_1)] \cap [P_2^{-1}L_m(G_2)] \tag{2.18}$$

ここで $P_1 \colon (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_1^*, P_2 \colon (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_2^*$.

2 つのオートマトンを同期合成したオートマトンと新しいオートマトンを同期合成することで 3 つのオートマトンの同期合成ができる．さらにこれを繰り返すことで 3 つ以上のオートマトンの同期合成をすることが以下に示す方法で可能である．

$$G_1 \mid\mid G_2 \mid\mid G_3 \mid\mid \cdots \mid\mid G_n = (\cdots((G_1 \mid\mid G_2) \mid\mid G_3) \mid\mid \cdots) \mid\mid G_n \tag{2.19}$$

2.3 スーパーバイザ制御

2.3.1 スーパーバイザと言語

スーパーバイザ制御は，制御対象，制御要求，スーパーバイザの 3 つの要素から構成されている．制御対象はオートマトンでモデル化されている．スーパーバイザは，事象の生起を禁止することによって，制御対象が制御要求を満たすように管理をする．

事象集合 Σ を 2 つの集合に分割して，可制御事象 $\Sigma_c \subseteq \Sigma$ と不可制御事象 $\Sigma_u \subseteq \Sigma$ とおく．Σ_c と Σ_u は共通する要素を持たず，ふたつの和集合は全体の事象集合 Σ と一致する． つまり次の式が成り立つ．

$$\Sigma = \Sigma_c \dot{\cup} \Sigma_u \tag{2.20}$$

スーパーバイザは可制御事象に対してのみ事象の生起を禁止することができ，不可制御事象の生起を禁止することはできない．そこで制御パターン γ という事象集合 Σ の部分集合を定義する．スーパーバイザが事象の発生を許可している制御パターン γ には次のような関係式が成り立つ．

$$\Sigma_u \subseteq \gamma \subseteq \Sigma \tag{2.21}$$

また，すべての制御パターンの集合 Γ を定義する．

$$\Gamma \coloneqq \{\gamma \mid \Sigma_u \subseteq \gamma \subseteq \Sigma\} \tag{2.22}$$

制御対象 G に対するスーパーバイザ制御 V は以下のように写像する．

$$V \colon L(G) \to \Gamma \tag{2.23}$$

事象列 $s \in L(G)$ に対して，それぞれに対応する制御パターン $V(s) \in \Gamma$ が存在する．

制御対象 G がスーパーバイザ制御 V の制御下にあるとき，V/G とかく．そしてそれを閉鎖的なフィードバック制御ループを図 2.1 のように示すことができる．

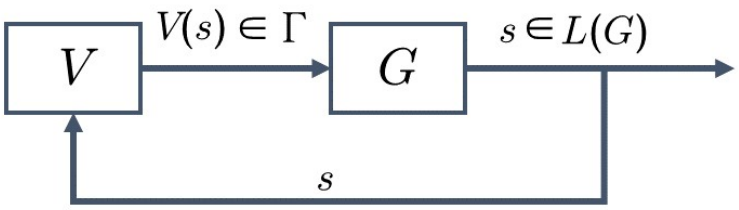

図 2.1　フィードバック制御ループ

1. G は事象列 $s \in L(G)$ を生成する．
2. s に対して，スーパーバイザ V が制御パターン $V(s) \in \Gamma$ を生成する．
3. 次の 2 つを満たした場合，G は s に続く σ を生成する．
   （i）$\sigma \in V(s)$（V に許可された σ）
   （ii）$s\sigma \in L(G)$（G で，s の後物理的に起こり得る σ）

4. $s\sigma$ を s に代入し，1 へ戻り繰り返す．

V/G によって生み出される言語は $L(V/G)$ とかき，以下のように定義する．

(i) $\epsilon \in L(V/G)$
(ii) $s \in L(V/G)\ \&\ \sigma \in V(s)\ \&\ s\sigma \in L(G) \Rightarrow s\sigma \in L(V/G)$
(iii) (i)，(ii) 以外の事象列は $L(V/G)$ に属しない．

また $L(V/G)$ について三式が成り立つ．

$$L(V/G) \neq \emptyset \tag{2.24}$$
$$L(V/G) = \overline{L(V/G)} \tag{2.25}$$
$$L(V/G) \subseteq L(G) \tag{2.26}$$

2.3.2 可制御性

制御要求を満たす言語 $K \subseteq L_m(G)$ をおく．V/G のマーク言語 $L_m(V/G)$ を以下のように定義する．

$$L_m(V/G) \coloneqq L(V/G) \cap K$$

また，このときの V を (K, G) に対するマーキングスーパーバイザ制御と呼ぶ．$\overline{L_m(V/G)} = L(V/G)$ のとき，V はノンブロッキングである．そして，K が制御可能のときかつその時に限り，ノンブロッキングである (K, G) に対する V が存在する．制御可能であるというのは，$\overline{K}$ に含まれるすべての事象列について

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K} \tag{2.27}$$

が成り立つときの K をいう．

制御可能であれば，制御要求を満たすスーパーバイザが存在する．つまりノンブロッキングなマーキングスーパーバイザを作ることができるかどうかは可制御性に依存している．

2.3.3 最大許容スーパーバイザ

K の制御可能な部分言語すべての集合を $C(K)$ とかく.

$$C(K) \coloneqq \{K' \subseteq K \mid \overline{K'}\Sigma_u \cap L(G) \subseteq \overline{K'}\} \tag{2.28}$$

また，以下のような式で表せる，$C(K)$ の要素のうちすべての K' を含んでいる最大要素 $supC(K)$ を定義する.

$$supC(K) \coloneqq \bigcup\{K' \mid K' \in C(K)\} \tag{2.29}$$

制御要求の言語 E ($E \subseteq \Sigma^*$) が与えられて $K = E \cap L_m(G) \subseteq L_m(G)$ とし，$K_{sup} = supC(K)$ ($\neq \emptyset$) であったとする．このとき,

$$L_m(V_{sup}/G) = K_{sup} \tag{2.30}$$

となる (K_{sup}, G) に対するノンブロッキングなマーキングスーパーバイザ制御 V_{sup} が常に存在する．またこのときのスーパーバイザを最大許容スーパーバイザとよぶ.

第 3 章

Human-in-the-Loop モデリング

この章では，本研究が着目する倉庫のモデル，およびロボットと人間のエージェントとしての定義をする．また，スーパーバイザの設計方法についても述べる．

3.1 倉庫のモデル化

倉庫の大きさや商品を保管する棚の配置など，倉庫によって構造が異なってくる．本研究では図 3.1 のような倉庫を考える．この倉庫の中で複数のロボットと人間がそれぞれの作業をするシナリオを考える．倉庫は，待機場所，通路，棚，搬出場所の 4 つの要素で構成されている．

上をロボットの待機位置，下をロボットが荷物を届ける搬出場所であると同時に人間が通常作業している場所である。上と下以外の色付きの部分を荷物の保管されている棚として，それら以外は通路とする．この通路はロボット一台が通れるくらいの幅で，すれ違うことは不可能であると想定する．

すべてのロボットにおいて，商品を積み込むときを除いて，いかなるときも棚に侵入できない．そして，棚に侵入するときは必ず上から入り下から出ていく．また，ロボットは前もしくは，右か左に進む動きしかできず，後ろ向きには進めないものとする．

ロボットは上の待機場所でタスクを割り当てられるのを待ち，割り当てられたら目的の品があるところまで行き，商品を積み，下の搬出場所までいく．ここでのタスクを割り当てられるというのは，スタート地点，ゴール地点，商品の場所の 3 つを与えられることで

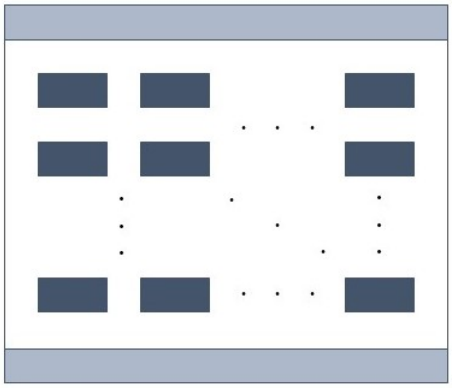

図 3.1　倉庫の構成

ある．スタート地点は待機場所の一段下の状態を，ゴール地点は搬出場所の一段上の状態を指す．商品の場所は棚のうちの一つを指す．

そして人間は、通常下で作業しており，トラブルが発生したときや，人間の協力が必要になったとき，ロボットも通る通路を使って目的の場所まで行く．また，人間はスーパーバイザの制御対象ではない．つまり，人間を制御できない対象として制御システム扱う必要がある．

また，倉庫内で複数台のロボットを制御するとき，以下の点において，注意する必要がある．

- 安全性：ロボット同士や人間とロボットの接触（衝突）を防ぐ
- デッドロック状態の回避：図のように，お互いのロボットが道を塞いでしまい，動けなくなる状態になることを回避する
- 効率性：すべてのロボットが荷物を届け終えるまでの時間が最短になるようにする

安全性については，ロボットだけの制御が場合，2 台のロボットが同じ状態に存在することを禁止する条件を，すべてのロボットの組み合わせで定めて制御要求として与える．スーパーバイザによってロボットの衝突を回避することが可能になる．しかし，この制御に人間が加わるとなると，より高度な安全性を考える必要がある．

デッドロック状態は図 3.2 のようにお互い進めなくなる状況である．これはノンブロッ

キングであることをスーパーバイザに要求することで解決される．

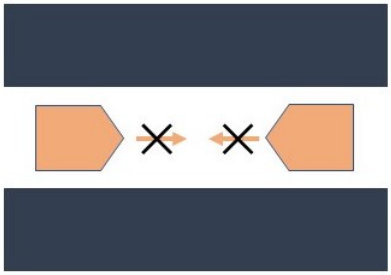

図 3.2　デッドロック状態

3.2　ロボットのモデル化

倉庫内のロボットはすべてオートマトンで設計する．第 2 章で紹介したオートマトンのように 5 つの要素で構成する $n(>1)$ 台のロボットを考えるとき，ロボットのオートマトンを以下の $G_i(i=1,2,\ldots,n)$ で定義する．

$$G_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i}) \tag{3.1}$$

ここで Q_i は i 台目のロボットの最短経路上の全状態の集合，Σ_i は i 台目のロボットの動作の集合，δ_i は状態遷移関数，$q_{0,i}$ は待機場所の状態，$Q_{m,i}$ は搬出場所の状態で構成されている．ロボットの動作は表 3.1 にあるように設定する．TCT では奇数の事象が可制御事象，偶数の事象が不可制御事象として処理される．ロボットの動作はすべて可制御事象なので奇数で与える．ここでは北，東，南，西に進むと記載しているが，図で示す場合，それぞれ上，右，下，左で表す．

事象	番号
北へ進む	$i \times 10 + 1$
東へ進む	$i \times 10 + 3$
南へ進む	$i \times 10 + 5$
西へ進む	$i \times 10 + 7$

表 3.1　i 台目のロボットの事象に振られる番号

3.3　人間のモデル化

倉庫内の人間も，ロボットと同様にオートマトンで定義する．人間が m 人倉庫内に立ち入るとする．ある人間 $H_j(j = 1, 2, \cdots, m)$ の普段の作業場所である一番下の位置（搬出場所）を初期状態 q_{0,H_j}，戻るべき目的地を Q_{m,H_j}，最短経路上の全状態の集合を Σ_{H_j} 状態遷移関数を δ_{H_j} とおいて，人間のオートマトンを以下で表す．

$$H_j = (Q_{H_j}, \Sigma_{H_j}, \delta_{H_j}, q_{0,H_j}, Q_{m,H_j}) \tag{3.2}$$

人間とロボットが存在する倉庫では，人間の安全面を絶対に確保することが求められ，最優先に考える必要があり，第 4 章で解説する．

3.4　スーパーバイザの設計

スーパーバイザを計算する手順について解説する．

本研究では TCT という離散事象システムの計算に特化したソフトウェアを使用し，計算を行う．

N 個の要素（エージェント）と K 個の制御要求が与えられたとする．

n 台のロボットの初期状態，受理状態，状態遷移をもとに，それぞれのオートマトン G_1 , $G_2, \cdots, G_n$ を作成する．また，人間のオートマトン $H_1, H_2, \cdots, H_m$ も作成する．n と m について，$n + m = N$ と書くことができる．

2 つのオートマトンに対して，2 つの状態で構成される状態のペアを与えられると，オー

トマトンそれぞれが状態のペアの状態に同時に存在しないようにする関数 mutex という関数がある．この関数で出力されるオートマトンを，制御要求のオートマトン E_1，E_2，$\cdots$，E_K とする．

次に，$G_1, \cdots, G_n, H_1, \cdots, H_m$ のすべてのオートマトンを同期合成させ，制御対象である PLANT を出力する．このとき，受理状態のみ存在し，PLANT のすべての事象で自己ループする遷移が定義されるオートマトン ALL を作成する．

E_1 から E_K までのすべての制御要求と PLANT の全事象を含んだ ALL を同期合成させ，制御要求 SPEC を生成する．

Supcon と呼ばれる関数に PLANT と SPEC を入力し，制御要求を満たすスーパーバイザを得る．

第 4 章

Human-in-the-loop における人間の安全性を確保する方法

この章では，人間を考えた倉庫の中で人間の安全性を確保するためにどう制御するかという問題に対しての 2 つの方法を提案する．人間の普段の作業場所をスタート地点といい，トラブルが発生した場所，もしくはロボットに協力するために向かうロボットとの合流地点を目的地，そして人間が最終的に戻りたい場所をゴール地点とよぶこととする．

4.1　方法 1

1 つ目の方法は，人間が使う通路上（スタート地点から目的地に向かうまでの経路と目的地からゴール地点に向かうまでの経路）をロボットに使わせないという考え方で制御を行う．人間が通る可能性のある通路についてロボットを完全に通行止めにする（図 4.1）．図の赤い部分がロボットの進入禁止エリアである．

これを実現する具体的な方法は，不可制御事象をもちいて人間の行動を定義する方法である．行動すべてを不可制御事象で定義するということは，スーパーバイザからみて，人間の行動がいつ発生するかわからない，かつ発生を禁止することができない．3.2 節のとおり，倉庫内を移動しているロボットはスーパーバイザ制御理論でいう可制御事象で定義されており，指示を出すことができる制御対象である．それに対して人間は制御対象ではない．

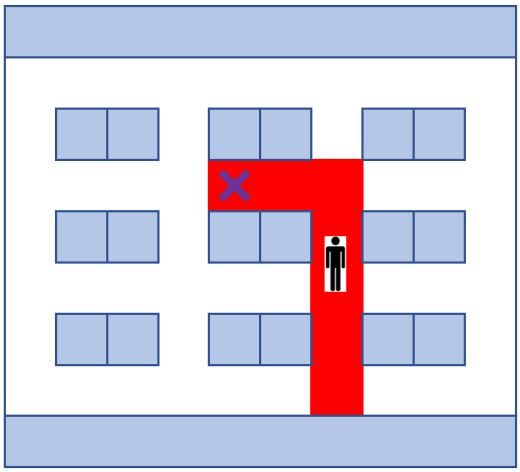

図 4.1　ロボットの進入禁止エリア

例えば，ロボットに対しては「特定の場所で方向転換し，あるタイミングで，決まった速度で移動する」というように精密に指示することが可能だが，人間はロボットのように厳密に制御できるわけではなく，本人の意思によっても行動する．したがって，人間を制御するのではなく，制御対象ではないとするのが自然である．

ロボット同士の制御であれば，制御要求を満たさない行動，つまり衝突してしまう行動やブロッキング状態になってしまう行動を直前で禁止することができた．しかし，不可制御事象で定義された人間に対しては，スーパーバイザは制御要求を満たさない行動を禁止できない．また，人間の行動のタイミングもコントロールするのが不可能である．ロボットが人間の経路上に存在していると，連続して何回も人間が行動し続けた場合，衝突することになり制御要求を満たさない．このような理由で人間の経路はロボットから見て障害物のように認識される．これを踏まえて，スーパーバイザが計算される．

この計算結果として，人間が安全な場所に到着するまで，人間の経路上にロボットを侵入させないというスーパーバイザが得られる．

言い換えれば，人間のスタート地点から，トラブルが発生した場所や協力が必要なロボットとの合流地点までの経路を予約し，予約された通路をロボットから見て通行止めとすることで，人間の経路上には完全に立ち入らせず安全を確保する方法である．

4.2　方法 2

2 つ目の方法は，人間のいる位置を重視して，人間の周りだけにロボットの進入禁止エリアを定めるというものである．1 つ目の方法は，ロボットの進入禁止エリアを静的に定めていたのに対し，2 つ目の方法では動的に変化する．

衝突回避の制御について，ロボット同士の場合は，どちらかのロボットの進行方向に別のロボットがいるときに行動を禁止している．これが人間とロボットの場合，人間の前に来てからロボットの行動を禁止するのでは，人間の恐怖心をあおるだけでなく，万が一エラーが起こることも考慮に入れると事故のもとになるという理由から，ロボットと人間の間に一定の距離を設ける必要性がある．

また，そのロボットと人間が安全を確保するために最低限離れていなければならない距離（最小安全隔離距離）を任意に決められるように変数で扱うことにより汎用性を高める．ロボットの移動速度が大きければ制動距離のことを考慮して最小安全隔離距離を大きくとり，逆に移動速度が小さければ最小安全隔離距離も小さくするというように，最小安全隔離距離を変数とし，変更可能にすることにより様々なスペックのロボットに対応することができる．また，倉庫の規模によって変更することもできる．

具体的な方法としては，制御要求を拡張することで実現する．ロボットだけを制御するとき，制御要求は衝突回避のため，同じ瞬間に同じ位置に 2 つ以上のエージェントが存在する状態を排斥していた．それに対し提案する方法では，最小安全隔離距離を 1 マスとおいた場合，図 4.2 のように制御要求を人間を中心としてロボットが前後左右のマスにいる状態にまで拡張させる．また，最小安全隔離距離を 2，3 とおいた場合，それぞれ図 4.3，4.4 のように進入禁止エリアを設定する．基本的に，ロボットが通路もしくは棚にいるときに接近を禁止する．ロボットが倉庫の待機場所もしくは搬出場所にいるときは距離をとる必要がないので進入禁止エリアに指定していない．

3.4 節の手順で計算を行うと，ロボットと人間がとり得るデータで制御要求を満たす状態遷移のすべてを網羅したスーパーバイザが出力される．そのスーパーバイザの状態遷移のうち，初期状態からどのようなふるまいで受理状態までの到達するのか選択する必要がある．すべてのエージェントが 1 回行動するもしくは行動しない一連の事象列の経過を単

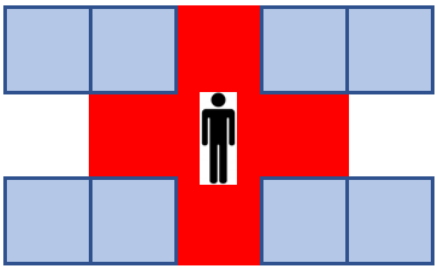

図 4.2　最小安全隔離距離 $d = 1$

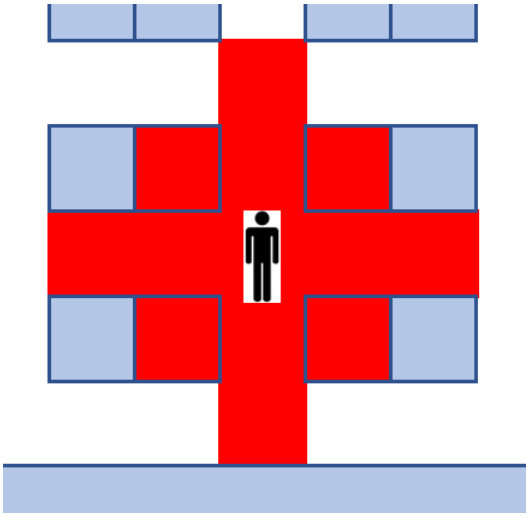

図 4.3　最小安全隔離距離 $d = 2$

位時間（スーパーバイザの対象の数が x とすると 単位時間 $< x$ 遷移）とする．ロボットのみの制御を行う場合，スーパーバイザの初期状態から受理状態までかかる時間が最小のものを選ぶことで最も効率の良いタスクの処理といえる．しかし，人間を含めた制御を行う場合，たとえタスクを完了させるまでの時間が短くても，安全性を考えると人間の行動を禁止してロボットを優先する制御をすべきではない．

また，実用化する観点からみたとき，ロボットのコストと人間のコストを比べると，ロボットを導入して運用することについては，運用費に対する初期投資が大きく，人を雇用

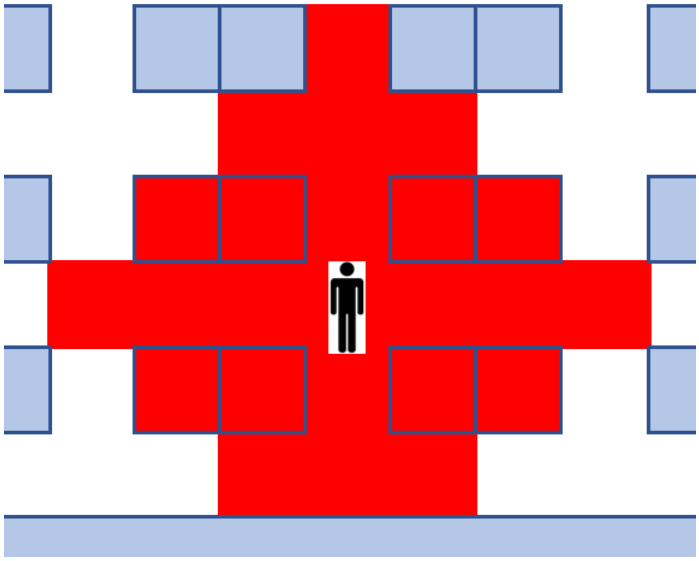

図 4.4　最小安全隔離距離 $d = 3$

するのについては，雇う時間が長ければ長いほどコストがかかると推測される．つまり，ロボットは固定費，人間は変動費といったように大まかに考えることにする．人件費においてはコスト削減の余地があるという点においても，ロボットと人間どちらかの行動を禁止するとなったとき，人間の行動を優先して，倉庫内の移動時間を短くすることが重要である．

第 5 章

倉庫で Human-in-the-loop した場合のシミュレーション検証

第 4 章で紹介した 2 通りの方法を用いて，人間が必要とされる具体的な事例をいくつか想定し，それらの方法を用いて問題解決できることを示す．2 台のロボットと 1 人の人間についてスーパーバイザ制御を行い，計算されたスーパーバイザをもとにシミュレーションを作成する．ロボット 1 とロボット 2 のオートマトンをそれぞれ G_1, G_2 とかき，人間のオートマトンを H とかく．

5.1　ケース 1：棚から商品が落下した場合

1 つ目のケースでは，人間の臨機応変な対処ができる能力を駆使し，発生の予測が不可能かつ早急に措置しなければならないトラブルに対処することが目的である．

まず，商品が棚から落ちた図 5.1 のような場合を初期状態と考える．2 台のロボットの初期状態は計算上，0 としているが，ロボット 1（$blue$）は状態 7 の上に，ロボット 2（$green$）は状態 8 の上に，一方人間の初期状態である通常作業場所は状態 67 の下に位置している．またそれぞれのタスクについて，ロボット 1 は状態 33（棚）に，ロボット 2 は状態 36（棚）に，人間はトラブルの発生した場所である状態 26（通路上）に設定する．搬出場所については計算上 71 とするが，G_1 は状態 68 の下に，G_2 は状態 65 の下に，人間 H はもとの作業場所であった状態 67 の下に戻ることとする．

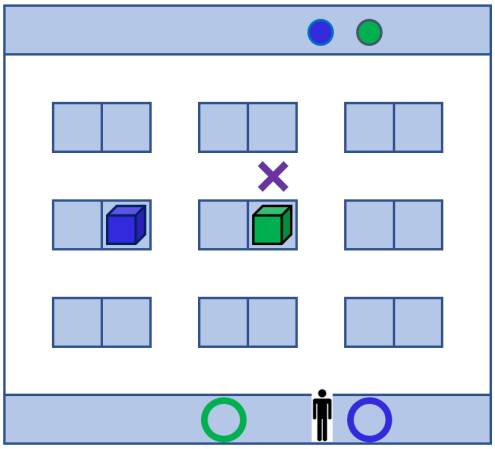

図 5.1　初期状態（ケース 1）

ケース 1 では，人間は棚から落ちた商品をもとあった収納スペースに返し，すぐ元の作業場所へ戻ってくると想定する．

ロボットのオートマトン G_1，G_2 と人間のオートマトン H の遷移関数はそれぞれの最短経路のみで定義するので，経路はそれぞれ図 5.2，5.3，5.4 とする．

落ちた商品で通路を塞いでしまうのでそのことに関しても考慮しなければならない．

このケースは，ロボット 1 ではトラブルの発生場所である状態 26 を通る経路がいくつかある最短経路のうちの 1 つとなっている．ロボット 1 に関して言えば，落下した商品

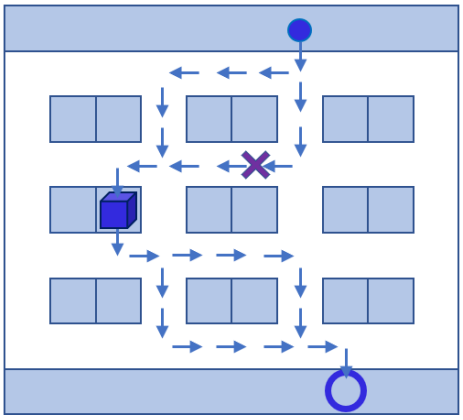

図 5.2　ロボット 1 の経路（ケース 1）

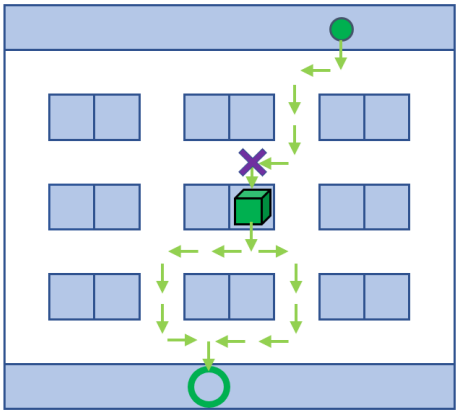

図 5.3　ロボット 2 の経路（ケース 1）

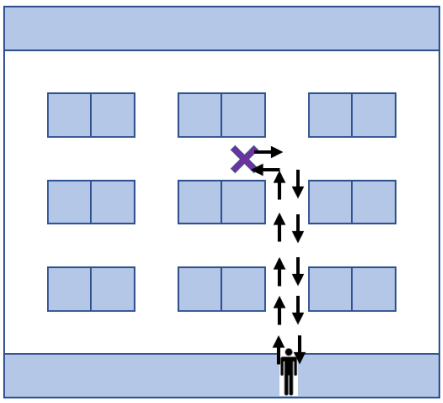

図 5.4　人間の経路（ケース 1）

で通路が使用できなくなっている経路を削除しても支障はない．しかし，ロボット 2 のオートマトン G_2 のようにトラブルの発生場所を使う経路しか定義されていないとき，そもそも受理状態まで到達できないという問題点が挙げられる．また，仮に搬出場所付近でこのトラブルが発生したとき，トラブルが処理されてすでに通ることができるにもかかわらず経路を削除してしまい効率の良いふるまいを見逃す可能性もある．落下した商品に関するオートマトンを作成し，それをもちいて制御要求を定めることで，これらの問題を解決する．

商品に関するオートマトン O は通路に落ちている状態 0 と棚に収納されている状態 1

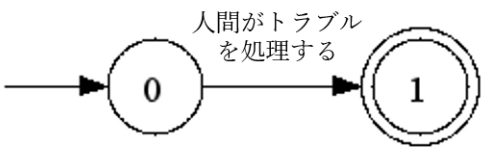

図 5.5　落下した商品についてのオートマトン

の 2 つの状態で構成されるオートマトンで，初期状態を 0，受理状態を 1 とする（図 5.5）．状態遷移は人間がトラブルを処理する事象（人間と共通の事象）が発生したとき，0 から 1 に移る状態遷移のみ定義する．ここで，指定した状態の組を排除する mutex という関数をもちいて制御要求を定める．

G_1 と O，G_2 と O について，mutex を使用し，状態の組み合わせ $P_{O,G_i} = [0, 26]$ を取り除く制御要求を与える． P_{O,G_i} の 1 つ目の要素は O の状態，2 つ目の要素は G_i の状態を示している．こうしてロボットの状態とトラブルが処理できていない状態の組み合わせを除くことで，商品が通路上にあるとき塞がった通路の使用を禁止し，かつ商品が棚に戻されたあとは通路の使用を許可する制御要求を与える．

H の状態遷移の定義について，スタート地点からトラブルが発生した場所へ向かう遷移と，その逆にトラブルの場所からもといた場所へ戻ってくる遷移があり，方法 1 では，これらはすべて不可制御事象によって遷移する．極端な話，1 マス進んで 1 マス戻るという遷移で受理状態に到達する可能性もあるが，スーパーバイザは行動を禁止することができずこれに対して対処できない．トラブルの発生場所を通る経路しか定義されていないロ

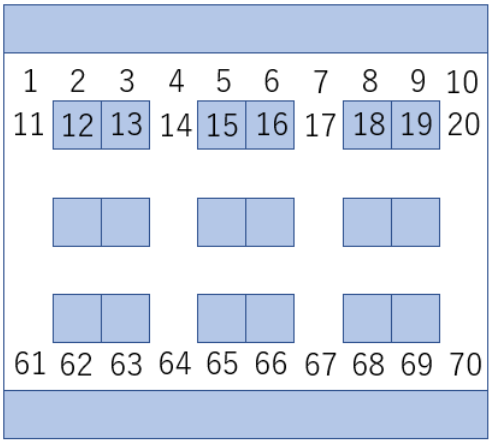

図 5.6　トラブルの処理前の状態に割り当てる番号

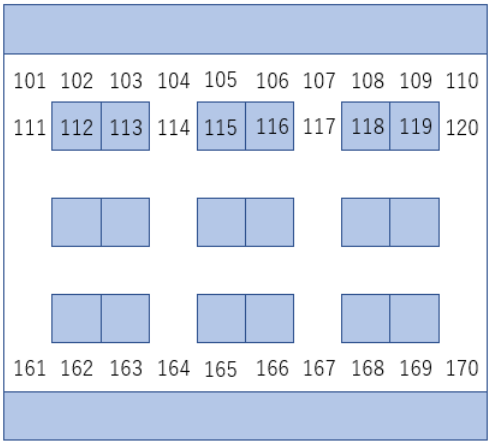

図 5.7　トラブルの処理後に状態に割り当てる番号

ボットが含まれていた場合，人間のトラブル処理の事象が起こるまで受理状態にいくことはない．しかし，先ほどの 1 マス進んで 1 マス戻る例では，人間がトラブルを処理せず受理状態に到達する可能性があるため，条件を満たすスーパーバイザが存在しないことになる．

人間の遷移が可制御事象で定義されていれば，オートマトン O を制御要求にすることにより，人間がトラブルを処理する前に受理状態に到達することがないように設定が可能

だ．しかし，人間の行動を不可制御事象にしてこの制御要求を与えても，条件を満たすスーパーバイザは存在しない．

これを解決するため，行きの状態と帰りの状態を区別することで，トラブルを処理する前に途中で引き返して受理状態にたどり着かないように設計した．状態の番号を図 5.6, 5.7 のように割り当てる．トラブルを処理する事象が起これば，帰り用の状態に遷移する．

商品で通路が塞がれているときはその場所を通る経路を削除し，人間がトラブルを処理したのち，削除した経路を追加しスーパーバイザの再計算を行うという方法も考えたが，現時点ではオンライン制御を実現できておらず，今後の課題として検討する．

5.1.1 方法 1

方法 1 の場合，人間の行動を表 5.1 のように定義することで，人間の経路はロボットからみてすべて通行不可となる方法であった．ここで，北，東，南，西に進む事象は，図で示すとそれぞれ上，右，下，左に進むことを意味する．トラブルを処理する事象は状態 s に存在したとすると，状態 s から状態 $s+100$ に遷移するという事象である．ロボットの進入禁止エリアは図 5.8 のように表され，人間が通る可能性がある限りロボットは進入できない．

事象	番号
北へ進む	100
東へ進む	102
南へ進む	104
西へ進む	106
トラブルを処理する	108

表 5.1　H の事象に振られる番号（ケース 1，方法 1）

行きと帰りの状態を分けることで起こってしまう問題がある．それは，トラブルを処理した後で，人間が通り過ぎたところからロボットが通路として使用するのが許可されることだ．作業場所に戻っている最中の人間のすぐ後ろをロボットが追尾する行動を許してしまうことになる．これでは，十分に安全ということはできない．

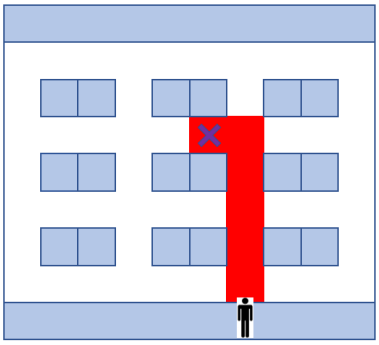

図 5.8　ロボットの進入禁止エリア（ケース 1，方法 1）

ここで新しいオートマトン P を定義する．図 5.5 のオートマトンのように，状態が 2 つ，遷移関数は初期状態から受理状態に遷移する事象ただ 1 つのみで定義する．H が受理状態にいるときに自己ループする事象を定義し，その事象と P の唯一の事象を一致させることによって，H が受理状態に到達した後で，P は状態 0 から状態 1 に遷移することになる．

そして，mutex を使用し，P が状態 0 に，G_1，G_2 が人間の経路上の状態に存在する組み合わせを排除する．これを制御要求に追加することで人間がもともといた作業場所に戻るまで，人間の経路に侵入するロボットの行動を禁止し続けることが可能である．オートマトン P は，人間が搬出場所に戻ったときのみ作動する，人間の安全を確認するスイッチのような働きをする．

G_1，G_2 と H について，衝突が起こらないようにそれぞれが同じ状態にいる状態を $PLANT$ から除く要求と，通路に落ちた商品が棚に戻されるまで，落ちた商品の位置にいる状態を除く要求を制御要求として与える．これらの制御要求をもとにスーパーバイザを作成し，フィードバックループ制御システムを構成する．

スーパーバイザによってどのような制御アクションを行われるかシミュレーションで解説する．

図 5.9 のような状態を状態 a と呼ぶことにする．状態 a のときロボット 1 は状態 7 に位置しており，ロボット 2 は状態 8 に位置している．このとき，次に生起するロボットの行動の選択肢として，G_1 が行動 15 によって状態 17 へ遷移，G_1 が行動 17 によって状態 6 へ遷移，G_2 が行動 27 により状態 7 へ遷移する 3 つが存在する．ここでロボット 2 が行

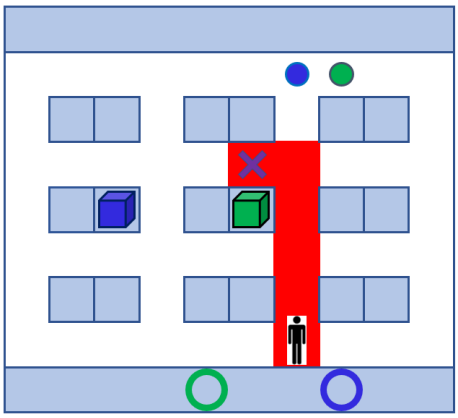

図 5.9　ケース 1 状態 a

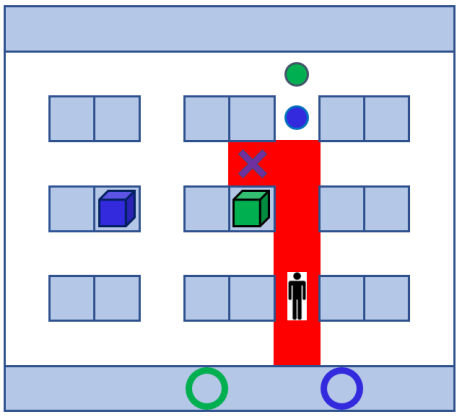

図 5.10　ケース 1 状態 b

動 17 で左に移動してしまった場合はロボット 1 と衝突することになり，制御要求を満たさない．そのため，このときスーパーバイザは G_2 の行動 27 を禁止する．このとき，その他の遷移はスーパーバイザに制御されることなく生起が許可されている．

G_1 は行動 15 を許可されているので，状態 17 に進み，その他のエージェント達も遷移して状態 a から状態 b（図 5.10）になったと仮定する．このとき状態 17 からの G_1 の遷移は行動 15 により状態 27 へ移る遷移だけである．しかし，状態 27 は人間の経路上に指定されているので，スーパーバイザが G_1 のこの行動を禁止する．人間がトラブルを処理してもとの作業場所に戻るまで，つまり H が状態 167（G_1 から見ると状態 67）から受理

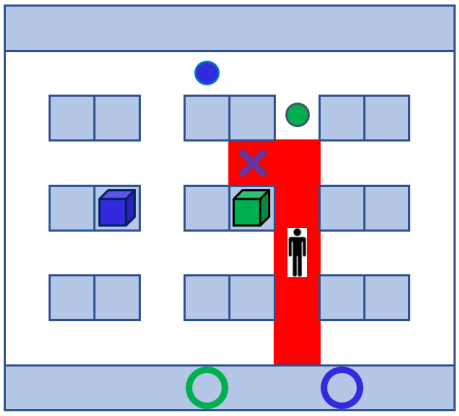

図 5.11　ケース 1 状態 c

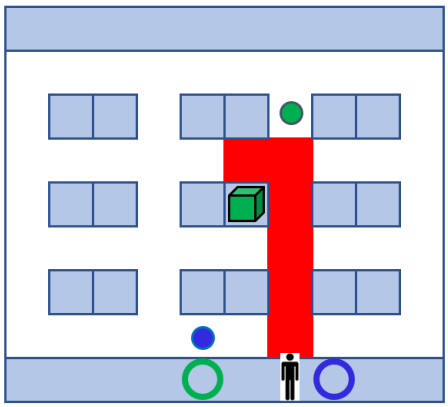

図 5.12　ケース 1 状態 d

状態に遷移して，受理状態での自己ループ事象が生起されるまで，G_1 の次の行動が禁止され続けることになる．

また，状態 a から，G_1 が行動 17，G_2 が行動 25，H が行動 100 が生起したときの状態を状態 c（図 5.11）とする．このときも状態 b のとき同様に，状態 17 にいる G_2 の遷移は行動 25 だけで，行動 25 により遷移した場合，人間の使用する通路に入ってしまう．制御要求を満たさないので，H が受理状態である 171 で自己ループするまで G_2 の行動 25 が禁止される．

H が受理状態 171 で安全が確保されたのち，図 5.12 のようにロボットの進入禁止エリ

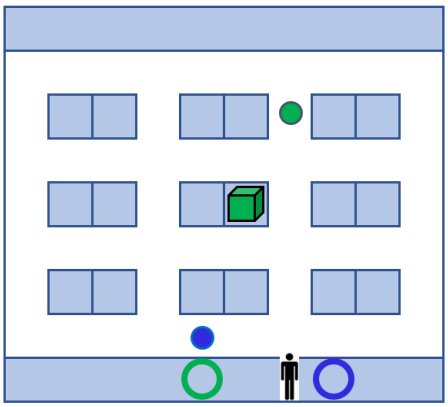

図 5.13　ケース 1 状態 e

アが解除され，対象がロボットだけの自動化制御が行われる．

5.1.2　方法 2

次に先ほどと同じ例を用いて，方法 2 によって制御した場合のスーパーバイザについて解説する．

まず，人間の行動を表 5.2 のように可制御事象で定義する．

事象	番号
北へ進む	101
東へ進む	103
南へ進む	105
西へ進む	107
トラブルを処理する	109

表 5.2　H の事象に振られる番号（ケース 1，方法 2）

状態 f（図 5.14）のとき，ロボット 2 のとり得る遷移は行動 25 である．しかし，このとき行動 25 により，状態 17 に遷移してその他のエージェントも遷移したとき，状態 g（図 5.15）になる．商品を棚に戻す作業をしない限り，ロボットは状態 26 に遷移できない

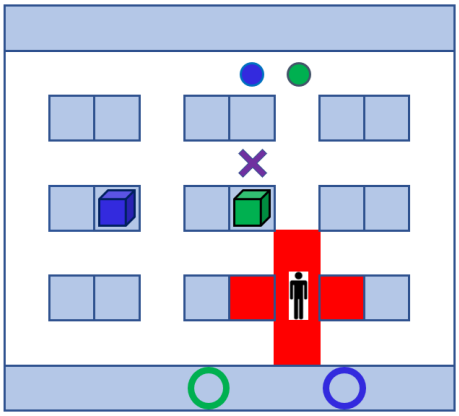

図 5.14　ケース 1 状態 f

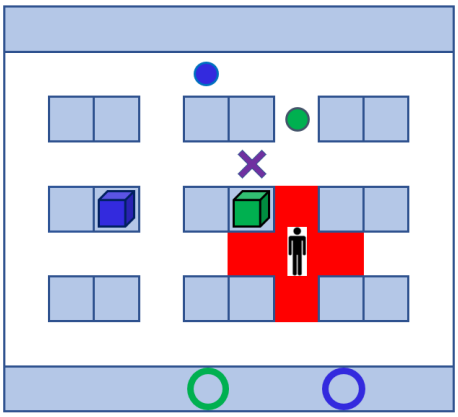

図 5.15　ケース 1 状態 g

制御要求があるのでロボット 2 は状態 27 に遷移したとしても受理状態まで到達することはない．また，人間とロボットの間に 1 マス以上間隔を開けなければいけない制御要求が与えられているため人間は状態 27 に遷移することができない．この結果，ロボット 2 と人間はお互い遷移ができないブロッキング状態に陥る．これを防ぐため，スーパーバイザは，状態 f で，G_2 の行動 25 を禁止する．

H が状態 137 に遷移した後（図 5.16），G_2 の状態 7 での行動 25 が許される．

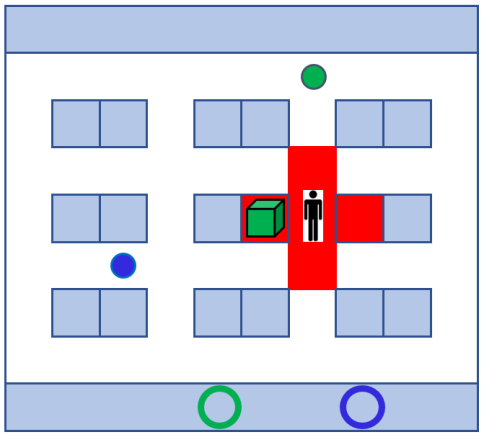

図 5.16　ケース 1 状態 h

5.2　ケース 2：人間とロボットが協調する

段ボール箱単体や，段ボール箱が乗ったパレットを運搬するような単純作業を行う場合，ロボットは能力を発揮する．それに比べて，商品のピッキング作業は不得意であり，今のピッキング技術では，商品の条件に変化があるような複数のタスクを 1 つのロボットで担うのは，困難であり，近い将来実現するのは難しいと考えられる．また人間は，ピッキング作業を得意としており，小さい品や，柔らかいもの，つぶれやすいものなど，さまざまな商品の条件に対応することができる．

荷物の運搬において，迅速に，かつ一度で大量に運ぶことも可能なロボットと，ピッキング作業において能力を発揮できる人間のそれぞれの長所を生かすことを目的として考えられた事例の制御システムを設計する．

ケース 2 では，具体的な例として図 5.17 を考え，ロボットだけでは完遂できない特殊なタスクが状態 32（棚）にあり，ロボット 1 に与えるとする．状態 4 の上をスタート地点とし，状態 64 の下に位置する搬出場所へ届けるというタスクが割り当てられている場合を想定する．また，ロボット 2 にはスタート地点を状態 6 の上に，搬出場所を状態 68 の下に，ピックする商品が状態 33（棚）にあるタスクを与える．人間は普段の作業場所を状態 66 の下とおいて，状態 31（通路上）に向かい，ロボット 1 の手助けをする．このと

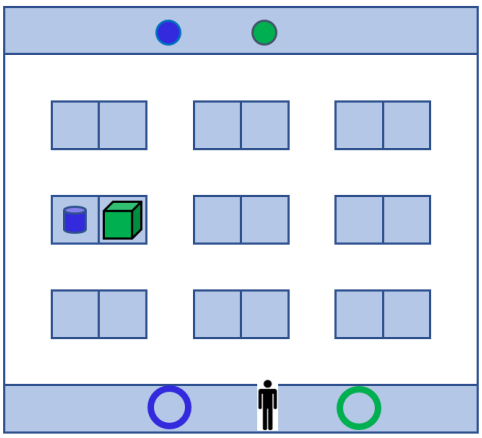

図 5.17　ケース 2 初期状態

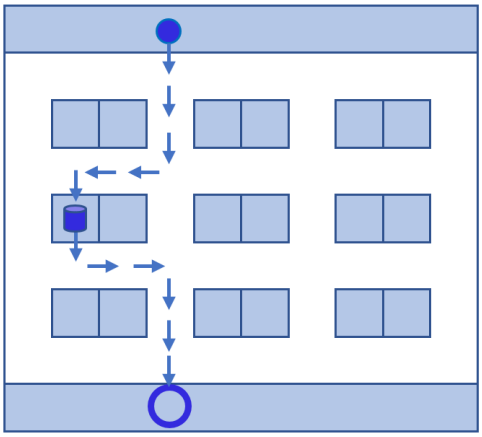

図 5.18　ロボット 1 の経路（ケース 2）

きの人間のポジションを協調作業場所と呼ぶことにする．ケース 1 と同様に，初期状態と受理状態を普段の作業場とする．

それぞれの最短経路は図 5.18，5.19，5.20 になる．

またケース 1 と同様に，図 5.6，5.7 のように行きの状態と帰りの状態を分ける必要がある．

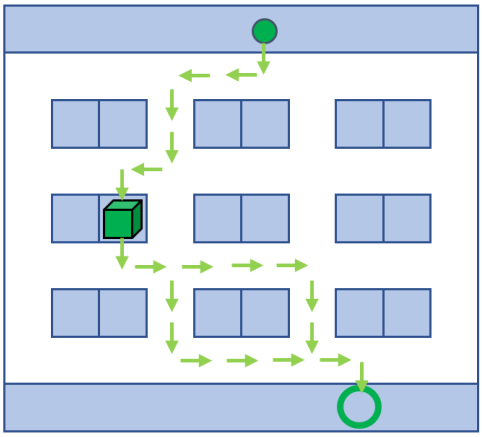

図 5.19　ロボット 2 の経路（ケース 2）

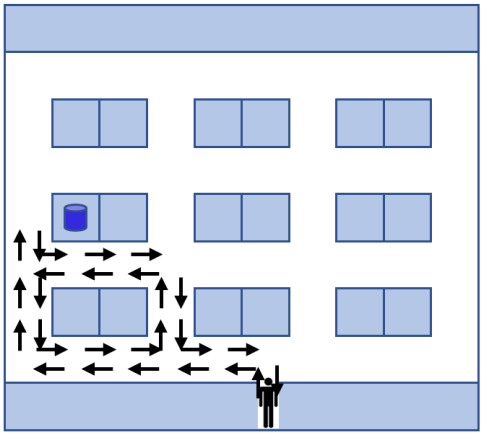

図 5.20　人間の経路（ケース 2）

5.2.1　方法 1

まず，人間の事象を表 5.3 のように設定し，オートマトン H を作成する．またロボットの進入禁止エリアは図 5.21 となる．

ロボット 1 は人間の手助けがないとタスクを完了できない．人間が状態 31 に，ロボット 1 が状態 32 に存在しているときのみ協調作業が可能である．人間が到着する前にロ

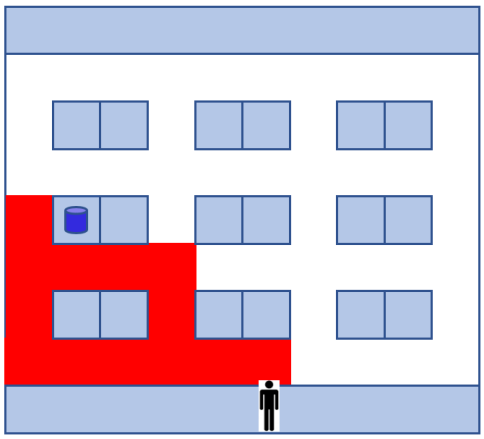

図 5.21　ロボットの進入禁止エリア（ケース 2，方法 1）

事象	番号
北へ進む	100
東へ進む	102
南へ進む	104
西へ進む	106
荷物をロボットに積む	108
受理状態での自己ループ	110

表 5.3　H の事象に振られる番号（ケース 2，方法 1）

事象	番号
北へ進む	11
東へ進む	13
南へ進む	15
西へ進む	17
人間に荷物を積まれる	109

表 5.4　G_1 の事象に振られる番号（ケース 2，方法 1）

ボット 1 が状態 32 から遷移することや，その逆にロボット 1 が到着する前に人間が状態 31 から遷移することを防ぐために，G_1 に H と共通の事象をおく（表 5.4）．この事象が生起すると G_1 は状態 32 で自己ループし，H は状態 31 から状態 131 に遷移する．

また，ケース 1 の方法 1（5.1.1 節）と同様に，H の受理状態で自己ループを行う事象を定義する．その H と共通の事象によってのみ状態遷移するオートマトン P を作成し，P をもとに制御要求を定めて，人間が安全な域に到達することが確認されるまで，進入禁止エリアを設ける．

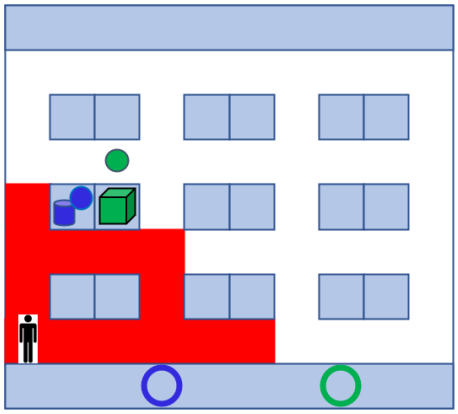

図 5.22　ケース 2 状態 a

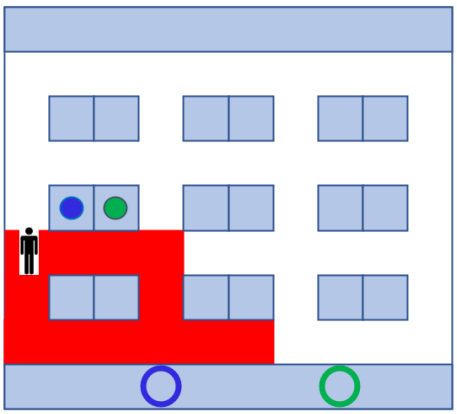

図 5.23　ケース 2 状態 b

図 5.22 のようにロボット 1 が状態 32 に存在するとき，ロボット 1 は行動 15 か行動 108 の遷移がある．行動 108 は人間，ロボット 1 が既定の状態に存在するとき以外起こりえない．また行動 15 は遷移先が人間の通路上の状態なので，スーパーバイザに禁止される．もし仮に行動 15 の遷移先が人間の経路上ではなかった場合でも，人間が状態 31 に到着したとき，ロボット 1 が状態 32 に存在せず，108 の生起条件を満たせない．つまり，受理状態に到達することがなくなるので，スーパーバイザが禁止する．

図 5.23 のとき（協調作業が完了した後），人間がもとの作業場所に戻っていく．ロボット 1，2 は遷移先が人間の経路で進入禁止エリアになっているので，人間のゴール地点で

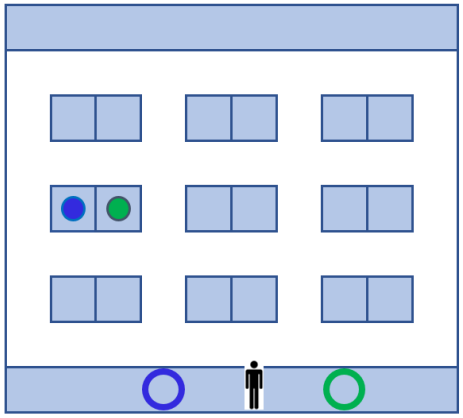

図 5.24　ケース 2 状態 c

安全が確認されるまで遷移は禁止される．

H が受理状態で自己ループする事象が生起したなら，スーパーバイザはロボット同士の衝突を回避する制御のみを行うようになる（図 5.24）

5.2.2　方法 2

ケース 2 でも，最小安全隔離距離は 1 マスとして，シミュレーションを行う．先ほどと同じ例を用いて，動的に進入禁止エリアが変化する方法 2 での制御を考える．この人間とロボットが協調する例を方法 2 で考えるとき，注意しなければならない点がある．ロボット 1 に割り当てられた荷物の位置にロボットが存在し，協調作業場所（この例では隣のマス）に人間が存在しているときのみ，協調作業が可能である．また，タスクを完遂するには協調作業をするために，人間と人間の協力を必要とするロボット 1 が，荷物の積み込みをする際に近づかないといけないのである．方法 2 の制御要求は，mutex という関数を用いて，同時に存在してほしくないロボットの状態と人間の状態の組み合わせを入力し，それらの状態の組を排除する制御要求を与えてスーパーバイザを作成する方法であった．この例では，人間とロボットの間に 1 マス以上の間隔をあける制御要求があるので，スーパーバイザを計算する際に与える制御要求について，人間がロボット 1 を手伝う作業場所に存在し，ロボット 1 が目的の商品の場所にいる状態の組みを削除したものを制御要求として与える必要がある．

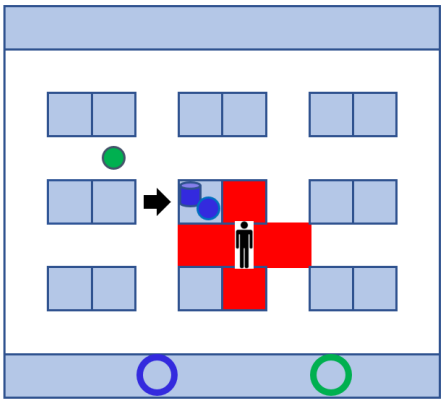

図 5.25　ケース 2 状態 d

しかし，このように制御要求の mutex に与える条件から 1 組だけ除くだけでは不十分であり，ブロッキング状態に陥る可能性がある．

次に，ブロッキング状態になる可能性がある問題の場面を，ケース 2 の例とは別の例を用いて紹介する．

図 5.25 のように，人間の助けを必要とするロボットが商品の棚の位置に，その隣に位置する協調作業場所に人間が向かっている場面を想定する．このとき，人間は協調作業場所へ向かうために左に移動しなければならない．しかし，左に遷移してしまうことで，ロボットと人間が接近しないという制御要求を満たさなくなる．そのため，スーパーバイザに人間が左に遷移する行動を禁止されてしまう．その結果，ロボットも人間も動くことができずブロッキング状態になる．人間がそこに行って荷物を積まないと受理状態まで到達できないということで図 5.25 の状態になる前に，ロボットが棚に移動する動作をスーパーバイザが禁止する制御をする．

また，その逆に人間が協調作業場所で待機しており，図 5.26 のように，ロボットが人間の周りのマスを経由して荷物の場所に向かっている場合についても同じ問題が生じる．ロボットの右に行く行動が禁止されてブロッキング状態になり，それを回避するため，スーパーバイザは人間が協調作業場所に到着する以前に制御を行うことになる．これは人間の行動を便宜上制御可能であると定義したことによって起こった問題である．

これらを踏まえると，ロボットと人間が協力して積まれる荷物周辺では，制御要求を緩和して，ロボットと人間の接近を許可する制御が求められる．つまり，協調して積み込む

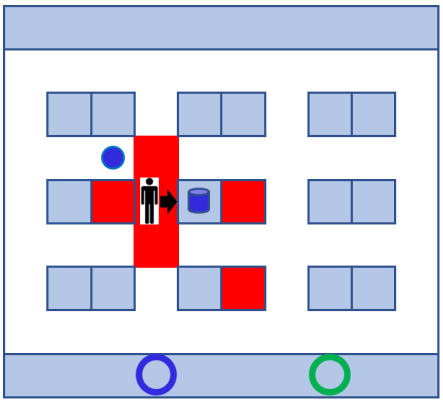

図 5.26　ケース 2 状態 e

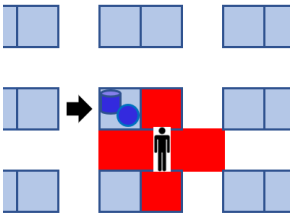

図 5.27　ロボットが待機しているときに発生し得るブロッキング状態

商品の保管場所近くでは，人間とロボットの間に距離をおくという制御要求の一部を削除する方法を提案する．具体的には，mutex の条件のうち，ロボットが商品の場所に存在している状態を含むペアを削除し，次に人間が協調作業場所に存在している状態を含むペアを削除するといった 2 つの手順を踏んで考える．

まずは，前者のロボット 1 が商品を保管する棚である状態 $[s_{G,item}]$ にいるときの状態の組を除外する．これはロボットが人間より先に協調作業場所に到着したときを想定した対策である（図 5.27）．

今回のシミュレーションでは，最小安全隔離距離を 1 マスとおいているので，人間側の状態に該当の位置の周りが含まれた状態のペアだけを削除すればよい．

商品のある棚の状態を $s_{G,item}$ とすると，削除の対象となるのは $[s_{G,item}, s_{G,item} - 10], [s_{G,item}, s_{G,item} - 1], [s_{G,item}, s_{G,item} + 1], [s_{G,item}, s_{G,item} + 10]$ である．$[s_{G,item}, s_{G,item}]$ も状態 $s_{G,item}$ を含んでいる．しかし，ロボットと人間が同じ状

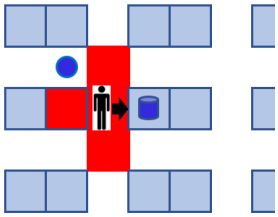

図 5.28　人間が待機しているときに発生し得るブロッキング状態

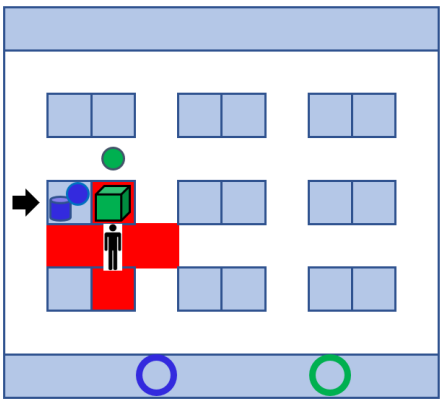

図 5.29　ケース 2 状態 f

態に存在する状態のペア $[s_{G,item}, s_{G,item}]$ は衝突回避の制御をする上で必要であるから，削除してはいけない．

もし，人間の安全を確保するために 2 マス以上確保しなければならないという制御要求を与えられていたなら，ロボット 1 が目的の商品の棚にいるときを含み，G と H が異なった状態にいるペアを削除することで解決される．

そして後者，人間が待機しているときロボットの経路に人間の周りが含まれていた場合も（図 5.28），前者同様に，ロボットと人間の状態が同じペアを除いて，人間が協調作業場所にいるすべての状態のペアを mutex の条件から削除する．

シミュレーションの結果である制御について解説する．

図 5.29 の状態のとき，ロボット 1 は荷物のある状態 32 に待機しており，ロボット 2 は状態 23 に，人間が状態 43 に存在している．このとき，人間に荷物を積んでもらうまでロボット 1 の次に発生する行動は禁止され続ける．ロボット 2 の遷移先は，人間が近くにい

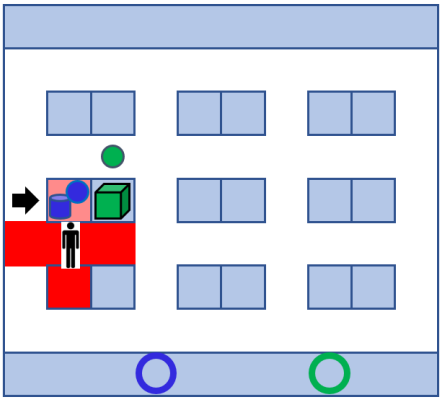

図 5.30　ケース 2 状態 g

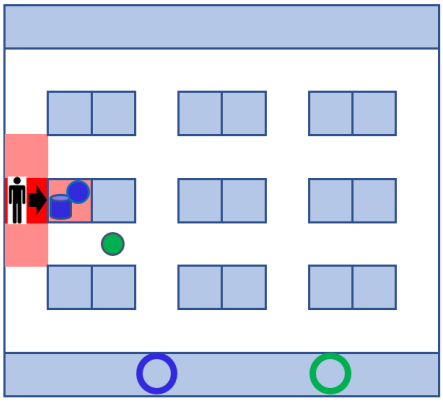

図 5.31　ケース 2 状態 h

るため進入禁止エリアとされているので，ロボット 2 の行動 25 が禁止される．また，H がとる可能性があるのは行動 107 である．ここで，ロボット 1 と人間に関する制御要求を緩和していることにより，人間の左へ遷移する行動が許されている．人間が左へ遷移したときの進入禁止エリアの変化は図 5.30 のようになる．図 5.30 の色の薄い進入禁止エリアはロボット 2 だけが対象であり，ロボット 1 には影響しない．

また，人間が協調作業場所に存在するとき，進入禁止エリアは図 5.31 のように変化する．

商品の積み込みの作業が終わり，人間が移動すると，再びロボット 1，2 は人間のまわ

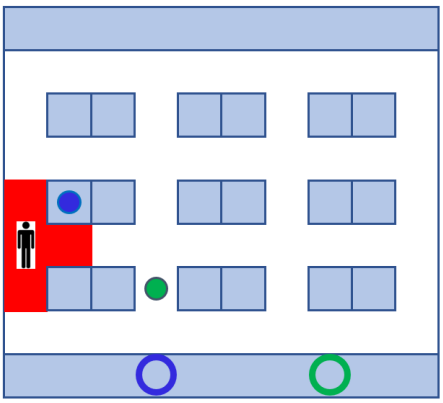

図 5.32　ケース 2 状態 i

りに接近を禁止されるようになる（図 5.32）.

5.3　2 つの方法の比較

方法 1 としてロボットに人間の経路上を使用させない方法，方法 2 としてロボットを人間に決められた距離より接近させない方法の 2 つの制御方法によるケーススタディを紹介した.

方法 1 では，人間の経路に指定されていれば，人間が離れていてもロボットが使用できず，別のルートを通るか待つ必要があった．方法 2 をもちいると，離れているときは，ロボットが待つことなく走行ができるというメリットがある.

しかし，方法 2 は方法 1 に比べて，経路が増加するたびに計算時間も増加する傾向がある（表 5.5）．また，最小安全隔離距離の d が増えるにつれ，さらに計算量が増加してしまう.

比較の表について，制御要求を作成して，スーパーバイザの計算が終わるまでの計算時間の 50 回の平均を示している．また，G_2 と H の状態遷移をそれぞれ経路の分岐が 1 つ増えるよう図 5.33，5.34 のように変更した.

	5.1 節の例	G_2 に遷移を追加	H に遷移を追加	G_2 と H に遷移を追加
方法 1	3.069	3.337	3.204	3.950
方法 2	3.222	3.974	4.105	5.098

表 5.5　計算時間の比較（単位：秒）

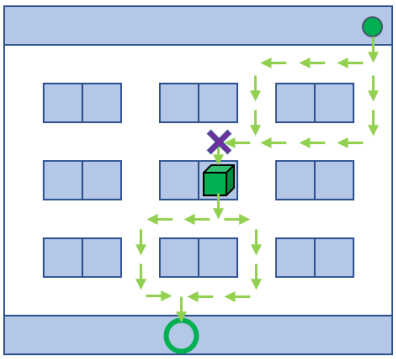

図 5.33　変更後の G_2 の経路

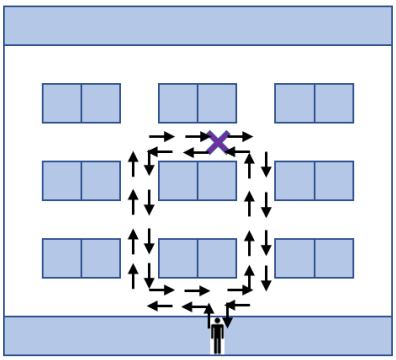

図 5.34　変更後の H の経路

第 6 章

結論

6.1　まとめ

離散事象システムのスーパーバイザ制御理論を用いて，倉庫の自動化を考える上で，倉庫内に人間が立ち入る際に人間の安全が保障されたロボット制御方法を 2 つ提案した．1 つは人間の使用する通路にロボットを侵入させない方法で，もう 1 つはロボットが決められた値以上人間に接近しないように，動的にロボットが入ることができない範囲を定める方法であった．

また，予期せぬトラブルが発生したときの対処と人間とロボットが倉庫内で協調作業をするケーススタディによって，提案した方法が有効であることを検証した．

そしてこれらの方法をもちいることで，人が倉庫内に行かなければならないとき，人間をシステムの一部として制御し，倉庫の自動化システムを稼働させたまま人間が立ち入ることができ，ロボットの無駄な待ち時間を削減し，作業効率の向上をもたらす．

6.2　今後の課題

オンライン制御をもちいて，トラブルをランダムなタイミングで発生させても対応できるように改良を加えると，より実用的なシステムになるので，オンライン制御での実装は今後の課題とする．

また，ロボット台数および人間の人数が少ないケースでは見えなかった問題が，エー

ジェントの数を増やすことによって浮き彫りになる可能性が考えられる．今回の研究では，ロボット 2 台，人間 1 人よりエージェントを増やしたシミュレーションを行うことができなかった．これは計算量が膨大になってしまい，今回使用した TCT の計算方法の限界であることが原因である．そこで Semi-Model Free[12] を用いることで問題が解決されると考えられる．

謝辞

本論文の執筆にあたり，多くの方々にご支援いただきました．中間報告会および審査会では，阿多信吾教授，岡育生教授，辻岡哲夫准教授，蔡凱准教授より，貴重なご指導とご助言を賜り，深く感謝申し上げます．特に，主指導教員である蔡凱准教授には，研究の進め方や枠組みにおいて，丁寧かつ熱心なご指導をいただきました．心より感謝申し上げます．

最後に，研究について有益なご討論，ご助言をいただきました同研究学科の皆様にお礼申し上げます．

ありがとうございました．

参考文献

[1] 総務省, **家計消費状況調査 ネットショッピングの状況について（二人以上の世帯）**—*2020* **年（令和** *2* **年）** *12* **月分結果**—, https://www.stat.go.jp/data/joukyou/12.html, 2021.

[2] P.J.Ramadge and W.M.Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[3] K. Cai and W. Wonham, *Supervisory Control of Discrete-Event Systems*. Springer, 2019.

[4] Y. Tatsumoto, M. Shiraishi, and K. Cai, "Application of supervisory control theory with warehouse automation case study," **システム／制御／情報**, vol. 62, no. 6, pp. 203–208, 2018.

[5] 松田裕貴, "都市環境分析におけるヒューマンインザループセンシングの研究," 奈良先端科学技術大学院大学／理化学研究所革新知能統合研究センター（AIP), Tech. Rep. 6, Aug. 2019.

[6] 神野悦太郎，小野瀬良佑，北栄輔, "SNS 投稿画像の注目度向上のための Human-in-the-Loop を用いたタグ推薦手法の検討," 名古屋大学大学院情報学研究科, 名古屋大学大学院情報学研究科, 名古屋大学大学院情報学研究科, Tech. Rep. 3, Sep. 2020.

[7] W. Li, D. Sadigh, S. S. Sastry, and S. A. Seshia, “Synthesis for human-in-the-loop control systems,” in *Tools and Algorithms for the Construction and Analysis of Systems*, E. Ábrahám and K. Havelund, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 470–484.

[8] I. Lage, A. Ross, S. J. Gershman, B. Kim, and F. Doshi-Velez, “Human-in-the-loop interpretability prior,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[9] **日本工業規格** *JIS D 6802*, 1997.

[10] 高井重昌, “離散事象システムのスーパバイザ制御,” **電子情報通信学会 基礎・境界ソサイエティ** *Fundamentals Review*, vol. 7, no. 4, pp. 317–325, 2014.

[11] 高井重昌，鈴木達也, “離散事象システム,” **計測と制御**, vol. 46, no. 4, pp. 248–254, 2007.

[12] Y. Tatsumoto, “A semi-model free approach for efficient supervisory control synthesis,” *Master’s Thesis, Osaka City University Graduate School of Engineering*, 2019.