

まだらの紐の真相を追って ～人工知能学会 推論チャレンジ問題～

2018年11月25日

株式会社野村総合研究所
田村 光太郎 外園 康智



〒100-0004
東京都千代田区大手町1-9-2 大手町フィナンシャルシティ グランキューブ

0. 差し替え部分

■10月31日送付の資料との変更部分

- ①本資料:15頁を追加。ALLOYと述語論理の関係の参考資料を追加。
- ②前資料:31頁を削除。(情報が重複していたため)
- ③場面3:殺人事件の真相の様相.alsを添付

■付記

- 29頁:1行目を付記
- 32頁:表の表現を修正

1. 応募者情報

■ 応募者の氏名、所属、連絡先

(代表)

● 田村光太郎

- 株式会社 野村総合研究所
 - (出向)NRIデジタル株式会社
- E-mail : k4-tamura@nri-digital.jp
- TEL : 070-4560-6025
- Web : <https://scholar.google.co.jp/citations?user=KHc7j4EAAAAJ&hl=ja>

● 外園康智

- 株式会社 野村総合研究所
- E-mail : y-hokazono@nri.co.jp
- TEL : 080-1353-6360

2. 推論・推理過程の説明

■目次

A) アプローチの特徴と結論

B) 解決アプローチ(手法とプログラム)

- ① オントロジーデータから、SPARQLで、主語述語目的語のRDFデータを抽出
- ② RDF→テンソルデータに変換
- ③ テンソルデータに対して、タッカー分解で、知識補完
- ④ テンソルデータに対して、推論規則を抽出、抽出した規則を使って、知識補完
- ⑤ CCG2lambdaにて、外部知識(日本語文)を述語論理式に変換
- ⑥ Alloyで3つの場面を設定。
 1. 形式手法Alloyの記述方法と知識の与え方
 2. 場面の区切りと設定
 3. 記述の内容
 4. 場面1:「ジュリア殺害の日」とAlloyによる結果
 5. 場面2:「ヘレン殺害未遂の日」とAlloyによる結果
 6. 場面3:「事件の真相の様相」とAlloyによる結果

C) まとめ

A)アプローチの特徴と結論

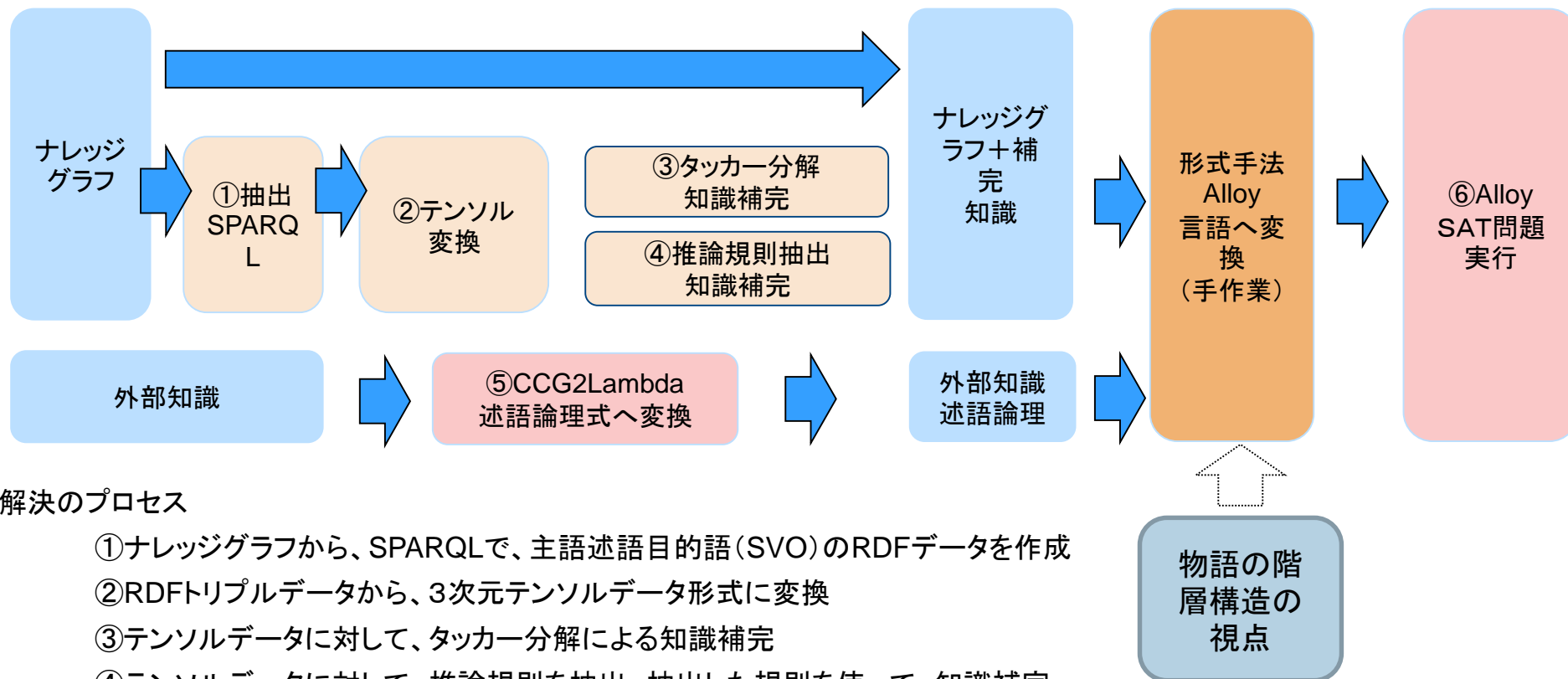
■アプローチの特徴

- 犯行状況を設定し、ナレッジグラフ(場面番号368番以下)＋外部知識を束縛条件に、述語論理式の充足可能問題として、真相を導く。
- テンソル分析で、知識補完を試みる。
- 物語の階層構造の視点を考慮する。

■結論

- 一連の事件(ジュリア殺害、ヘレン殺害未遂)は、ロイロットが犯人。
- 一部事件(ロイロット殺害)は、ヘレンが犯人とも推察されるが、証拠不十分。

B)アプローチ(プロセスとプログラム)



■ 解決のプロセス

- ①ナレッジグラフから、SPARQLで、主語述語目的語(SVO)のRDFデータを作成
- ②RDFトリプルデータから、3次元テンソルデータ形式に変換
- ③テンソルデータに対して、タッカー分解による知識補完
- ④テンソルデータに対して、推論規則を抽出、抽出した規則を使って、知識補完
- ⑤CCG2Lambdaにて、外部知識を論理式に変換
- ⑥ALLOYで3つの状況を設定。

場面1： ジュリア殺害の日

場面2： ヘレン殺害未遂の日

場面3： 殺人事件の真相の様相

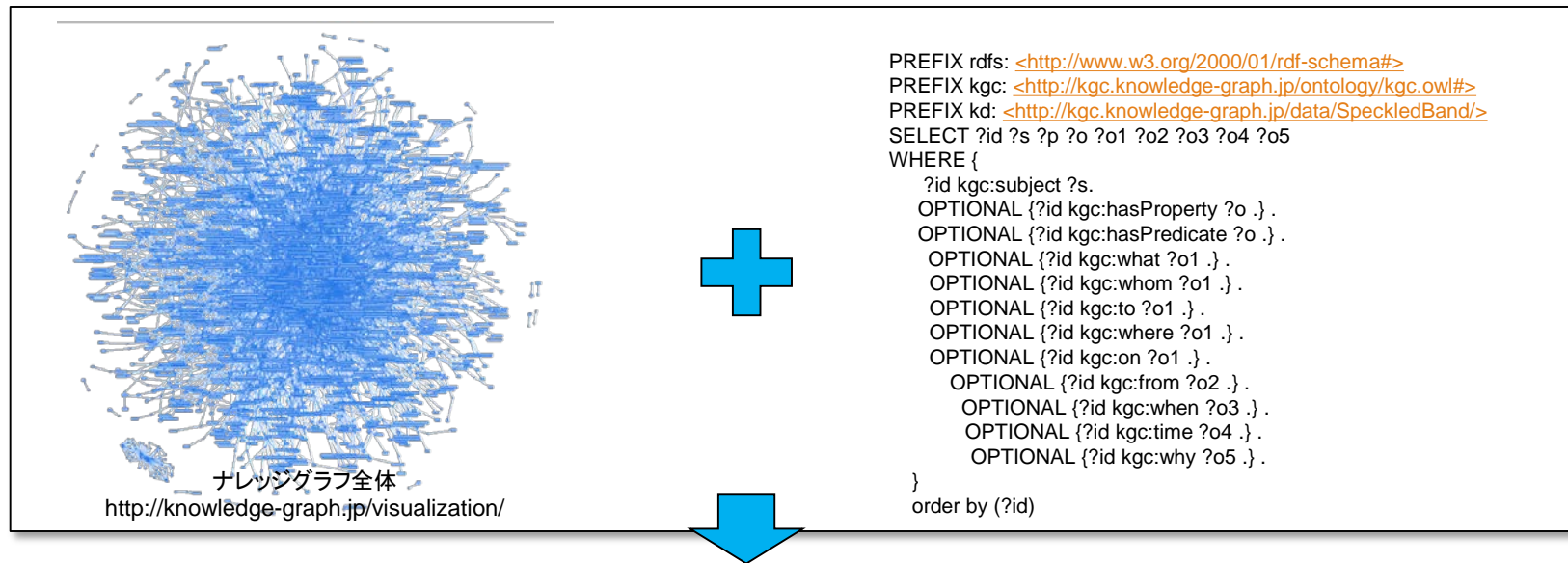
①オントロジーデータから、SPARQLで、主語述語目的語のRDFデータを抽出

■ 手順の目的

- 関係性解析をテンソル形式で行うため、ナレッジグラフを変換する。

■ 手順の概要

- 以下のSPARQLファイルで、主語述語目的語、場所FROM、場所TO、時間、理由 データを抽出



場面	述語	主語	対象	場所	起点	終点	時間	何	理由	どうして
1	come	Helen	NA	NA	NA	house of Holmes	NA	NA	NA	NA
2	beScared	Helen	NA	NA	NA	NA	NA	NA	NA	NA
3	beUpset	Helen	NA	NA	NA	NA	NA	NA	NA	NA
4	notHave	Helen	NA	NA	NA	NA	NA	money	NA	NA
5	getMarried	Helen	NA	NA	NA	NA	within 2 months	NA	NA	NA
6	obtain	Helen	NA	NA	NA	NA	within 2 months	money	NA	NA
7	pay	Helen	Holmes	NA	NA	NA	NA	money	reward of request	NA
8	live	Helen	NA	mansion of Roylott	NA	NA	NA	NA	NA	NA

②RDF→テンソルデータに変換

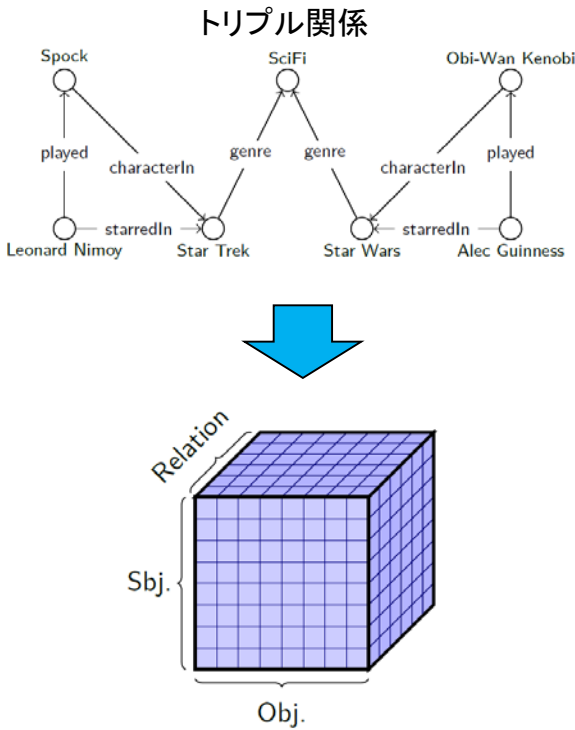
■ 手順の目的

- SVOOまたはSVOとして書き表された、RDFトリプルをテンソルに変換する。

■ 手順の概要

- 前項抽出されたデータをSVO形式(SVO_1O_2 は SVO_1 と SVO_2)に変換し、テンソル形式にする。
- 疎テンソルであるため隣接リスト表現を利用した。

主語	述語	目的語(補助)
Crown_Inn	exist	exist
Crown_Inn	be	hotel
Helen	beScared	bedroom_of_Julia
Helen	beScared	exist
Helen	beUpset	exist
Helen	call	Roylott
Helen	cannotFind	objects
Helen	change	clothes
Helen	come	house_of_Holmes
Helen	come	Leatherhead_station
Helen	come	bedroom_of_Julia
Helen	exist	bedroom_of_Julia
Helen	getMarried	exist
Helen	go	Crown_Inn
Helen	go	Leatherhead_station
Helen	go	Stoke_Moran
Helen	go	bedroom_of_Helen
Helen	go	bedroom_of_Julia
Helen	have	two-wheeled_coach



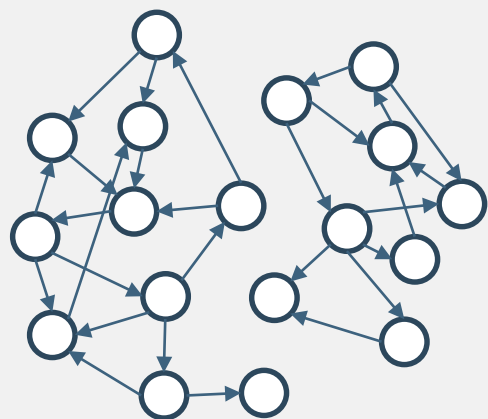
③、④テンソル解析を利用した知識補完の考え方

■ 与えられたナレッジグラフから知識を補完・抽出する

- 不完全なナレッジグラフに対して、連結性を高めるために欠損リンクの補完
- ナレッジグラフ上の探索で未知の規則の抽出

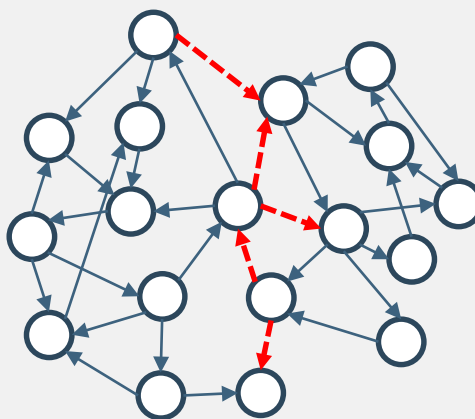
②ナレッジグラフ(テンソル)

トリプル化されたナレッジグラフ



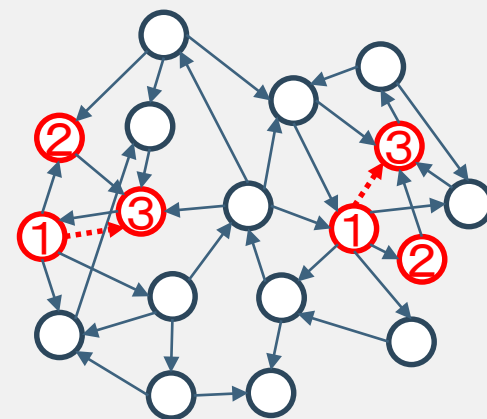
③Tucker分解による知識補完

途切れている知識関係を補完する



④探索による推論規則抽出

連鎖する関係から推論規則(同様の関係なら起こる確率が高い)を抽出



※改善中

③テンソルデータに対して、タッカー分解で、知識補完

■ 手順の目的

- ナレッジグラフから推測される尤もらしい事実(知識)を数理的手法により抽出する。
- 知識補完を行うことで、外部知識の補完の補助ができるかの検証。

■ 手順の概要

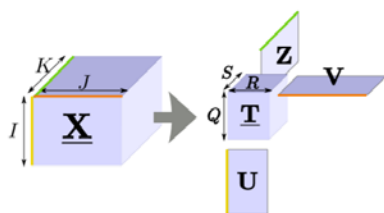
- テンソルデータの欠損補完の主な方法であるテンソル分解→再計算でテンソルを補完する。

■ タッカー分解

- 3次元のテンソルXをコアテンソルTと因子行列U,V,Zに分解する。
- 分解後のコアテンソルTと因子行列U,V,Zで、もとのテンソルを再計算する。
- コアテンソルTと因子行列U,V,Zから、与えられたテンソルデータに対して、モデル(尤度を最大)となるテンソルが得られる。

Tucker 分解 [Tucker 1966]

X をコアテンソル T と因子行列 U, V, Z に分解



$$x_{ijk} \simeq \sum_{q=1}^Q \sum_{r=1}^R \sum_{s=1}^S t_{qrs} u_{iq} v_{jr} z_{ks}$$

- U, V, Z : 各モードの基底
- T : 座標

T が超対角のとき PARAFAC と一致

Tucker 分解の確率的解釈

簡単のため2次のテンソル(=行列)の場合を考える

$$\mathbf{X} = \mathbf{U} \mathbf{T} \mathbf{V}^T + \mathbf{E}$$

- $E_{ij} \sim N(0, \sigma^2)$: 観測ノイズ

このモデルの対数尤度は Tucker 分解の目的関数と等価

$$\ln p(\mathbf{X} | \mathbf{U}, \mathbf{V}, \mathbf{T}) = \frac{1}{2\sigma^2} \|\mathbf{X} - \mathbf{Y}\|_{\text{Fro}}^2 + \text{const.}$$

where $\mathbf{Y} = \mathbf{U} \mathbf{T} \mathbf{V}^T$

- \mathbf{Y} : \mathbf{X} の背後にある真の行列

③テンソルデータに対して、タッカー分解で、知識補完

■ テンソルの補完のための計算の条件

計算量が大きいためデータに制限を設けた

- Subjectが人物、動物の場合に絞ったテンソルを作成し、Tucker分解
- 18(人物と動物) × 146(動作) × 18(人物と動物)のテンソル
 - ・ 機械的に取り出したので、同一人物を指す単語が含まれている。
 - ・ Animalという項目をLeopard, baboon, mouse, cat

知識補完のために設定したパラメータ

- Tucker分解: コアテンソル(3 × 15 × 3)を使って、復元
- 複数回繰り返し、復元されたテンソルで、要素が成分1(>0.5)として復元された部分を補完とみなした
- 補完された知識を初期テンソルに加えて、Tucker分解→知識補完を繰り返した

対象の登場主体

Helen
Holmes
Julia
Leopard
Roma
Roylott
Watson
baboon
cat
coroner
craftsman
father-in-law
housekeeper
man
mouse
sister
suspect
Animal

③テンソルデータに対して、タッカー分解で、知識補完

■ Tucker分解に利用したPythonスクリプト

- [Tucker.py : http://yamaguchiyuto.hatenablog.com/entry/2016/11/30/080000](http://yamaguchiyuto.hatenablog.com/entry/2016/11/30/080000)

```
import pandas as pd
import matplotlib.pyplot as plt
from Tucker import Tucker

tens = pd.read_table('tensor_focused.dat',header=None,sep=' ')
arr = [[[0 for i in range(18)] for j in range(147)] for k in range(18)]
arr = {}

for i in range(18):
    for j in range(147):
        for k in range(18):
            arr[(i,j,k)]=0
for i in range(len(tens)):
    arr[(tens[0][i],tens[1][i],tens[2][i])]=1

model = Tucker(R=3,S=15,T=3,max_iter=10)

model.fit(arr)
print("END")
residual=0
count=0
for i in arr:
    residual += abs(arr[i]-model.predict(i))
    count += 1
print(residual/count)

plt.plot(model._losses)
plt.ylabel('Loss')
plt.xlabel('# iters')
plt.title('Tucker')
plt.show()
for i in arr:
    a = model.predict(i)
    if arr[i] == 0 and a > 0.1 :
        print(i,a,arr[i])
```

ライブラリのインポート

※上記のTucker.pyを3行目で読み込んでいる

テンソルの読み込み

tensor_focused.datと名前をつけたファイルを読み込み、arrの要素に代入する。

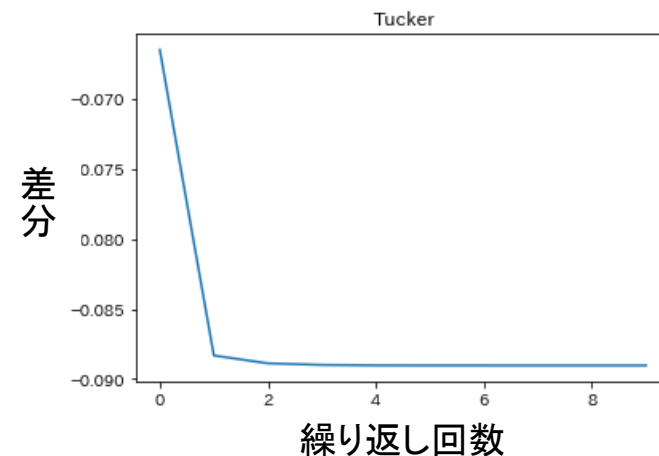
3 × 15 × 3のコアテンソルを作るように分解する。
繰り返し計算での収束性を見るために

分解したコアテンソルと因子行列を用いて計算し、
もとのテンソルと成分を比較し、補完されている要素(知識)を抽出する。

③テンソルデータに対して、タッカー分解で、知識補完

■ 計算の妥当性

- 与えたコアテンソルの大きさ($3 \times 15 \times 3$)、繰り返し回数(10回)
- 各繰り返し計算でのコアテンソルの変化の収束性については、10数回程度で十分小さく、収束にいたっていると考えられる。
(※n回目とn-1回目のコアテンソルの要素の差分)
- 疎テンソルのため、あまりアルゴリズムが順応していないのか差分が比較的多く残る。



■ 補完された知識(※場面は367番までの情報。)

- 行列が低ランク(同じ行や列がしやすいように)になるように補完される傾向がある。
 - ・ 行動が似ている人(ワトソン、ホームズ)同士の間で、片方しか行っていない行動があるとき、他方にその行動をしたかのような補完がなされた
- 補完された知識のなかには解釈が困難なものもあるが、Juliaに対して、Animalが何らかの行為を行ったことが多数出る。
- 補完された知識
 - Homes See Animal
 - Watson Hit Animal
 - Watson Put Animal
 - Homes Go Royslott
- 以下、成分0.3程度だができたこと
 - Helen bite Julia
 - Animal Call Julia
 - Animal Meet Julia
 - Animal Support Royslott
 - etc

④テンソルデータに対して、推論規則を抽出、抽出した規則を使って、知識補完(改善中)

■ 手順の目的

- 補完されたテンソルデータをもとにして、推論規則の抽出(さらなる知識の補完)を行うことでテンソルを密にする。

■ 手順の概要

- 先述の補完されたテンソルをもとに、幅優先探索を用いて、テンソルネットワークの2リンク関係、3リンク関係、～を抽出する。

■ 推論規則

- 推論規則であるためにPerson、Objectsなどのラベルで表す。
 - ・ 例 : Person Call Person、Person Pull Objectsなど
 - ・ 例(右下) : $a \rightarrow c$ までのパス全てを列挙し、パスの有意性を閾値を持って確かめる
- 数が少ないのと時間を考慮することが難しい
- 物語中、1回しかでてこない関係を規則としてしまう

推論規則の抽出 [YYH⁺14]



▶ 推論規則

$$\text{bornInCity}(a, b) \wedge \text{cityOfCountry}(b, c) \Rightarrow \text{nationality}(a, c)$$

- ▶ 新しい Fact の生成
- ▶ 知識グラフのコンパクトな格納
- ▶ 高度な推論システム

▶ 長さ k の規則を抽出 (実験では $k = 2, 3$ に限定)

- ▶ Relation r に対し, \mathcal{X}_r と \mathcal{Y}_r を Subject, Object 候補の集合
- ▶ 開始 Relation 集合 $\mathcal{S} = \{s : \mathcal{X}_s \cap \mathcal{X}_r \neq \emptyset\}$
- ▶ 終端 Relation 集合 $\mathcal{O} = \{t : \mathcal{Y}_t \cap \mathcal{Y}_r \neq \emptyset\}$
- ▶ \mathcal{S} と \mathcal{O} から可能な Relation path r_1, \dots, r_k を列挙
- ▶ $\mathbf{y}_a^T \mathbf{W}_{r_1} \approx \mathbf{y}_b$ 及び $\mathbf{y}_b^T \mathbf{W}_{r_2} \approx \mathbf{y}_c$ から $\mathbf{y}_a^T \mathbf{W}_{r_1} \mathbf{W}_{r_2} \approx \mathbf{y}_c$

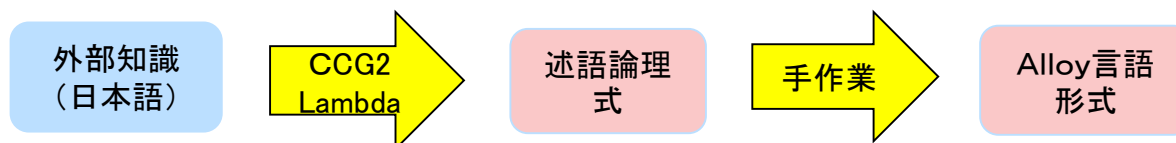
$$\text{dist}(\mathbf{W}_r, \mathbf{W}_{r_1} \dots \mathbf{W}_{r_k})$$

で閾値を越える推論規則を抽出

⑤CCG2lambdaにて、外部知識(日本語文)を述語論理式に変換

目的と手順の概要

- 外部知識(日本語文)を、プログラムで推論しやすい述語論理式に変換する。



例1

- ① 全ての人は、自分を殺さない。
- ② $\exists x. (\exists z1. (\text{全て}(z1) \ \& \ (x = z1)) \ \& \ \text{人}(x) \ \& \ \neg \exists z1. (\text{自分}(z1) \ \& \ \exists e. (\text{殺す}(e) \ \& \ (\text{Nom}(e) = x) \ \& \ (\text{Acc}(e) = z1))))$
- ③ $\text{fact}\{\text{all } p:\text{Person} | \text{all } t:\text{Time} | \text{all } r:\text{Room} \{ \text{no } p.\text{kill}.t.p.r \} \}$

例2

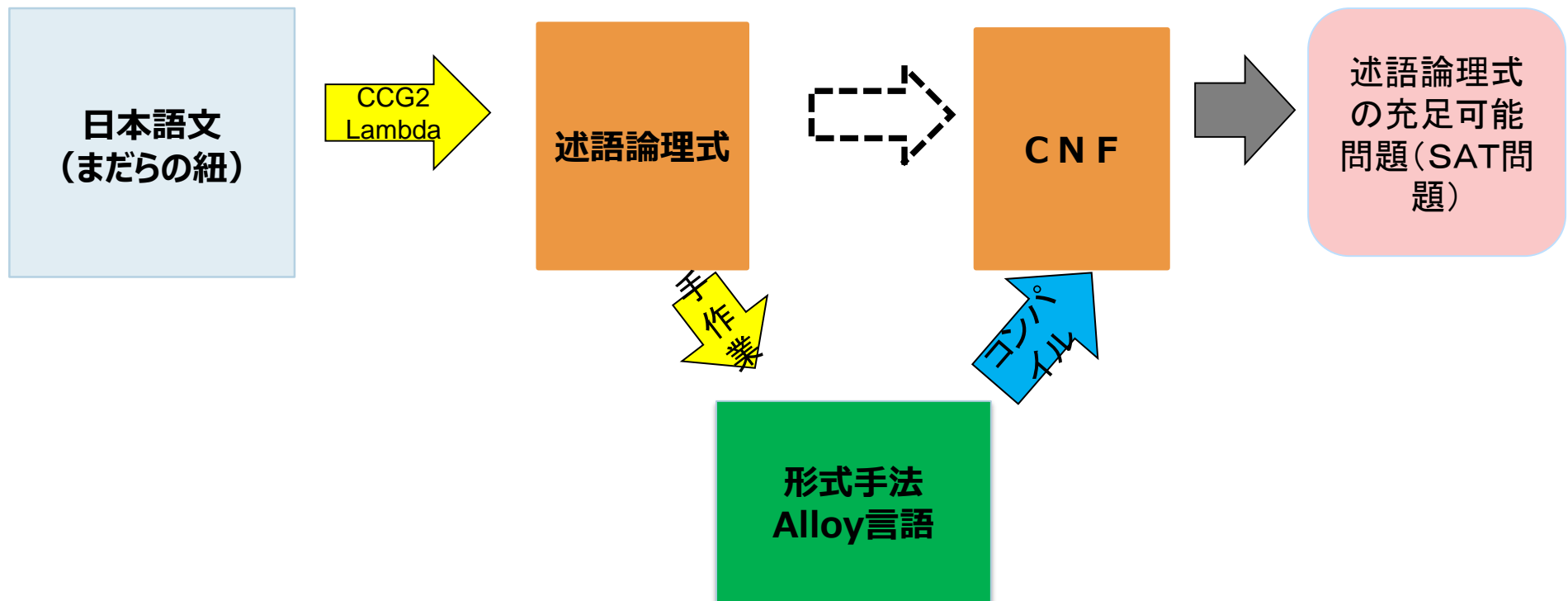
- ① ある人が、毒を盛られたならば、ある人は死ぬ。
- ② $(\exists x. (_人(x) \ \& \ \exists z2. (_毒(z2) \ \& \ \exists e. (_盛る(e) \ \& \ \text{Past}(e) \ \& \ \exists z2. (_人(z2) \ \& \ (\text{Dat}(e) = z2)) \ \& \ (\text{Acc}(e) = x) \ \& \ (\text{Acc}(e) = z2)))) \rightarrow \exists x. (_人(x) \ \& \ \exists e. (_死ぬ(e) \ \& \ (\text{Nom}(e) = x))))$
- ③ $\text{all } q:\text{Person} | \text{all } t:\text{Time} \{ \text{some Person. 毒を盛る}.t.q \Rightarrow \text{dead}[q,t.\text{next}] \}$

注 述語論理式からAlloy言語に変換するプログラムはできておらず、手作業で変換。

ただし、日本語文→述語論理式→CNF(連言標準形)→充足可能問題(SAT問題)が正しい流れで、AlloyはCNFへのコンパイラ。

参考) 日本語文と述語論理式と形式手法Alloy言語の関係

- 日本語文→述語論理式→CNF(連言標準形)→充足可能問題(SAT問題) が本来の流れである。
- AlloyはCNFへのコンパイラである。
- ただし、オブジェクト定義やリスト構造など、Alloy言語仕様により、述語論理式同士の記号の共通化や、などが図れるため、述語論理式を直接、SAT問題に直すより、問題の見通しがよくなる。



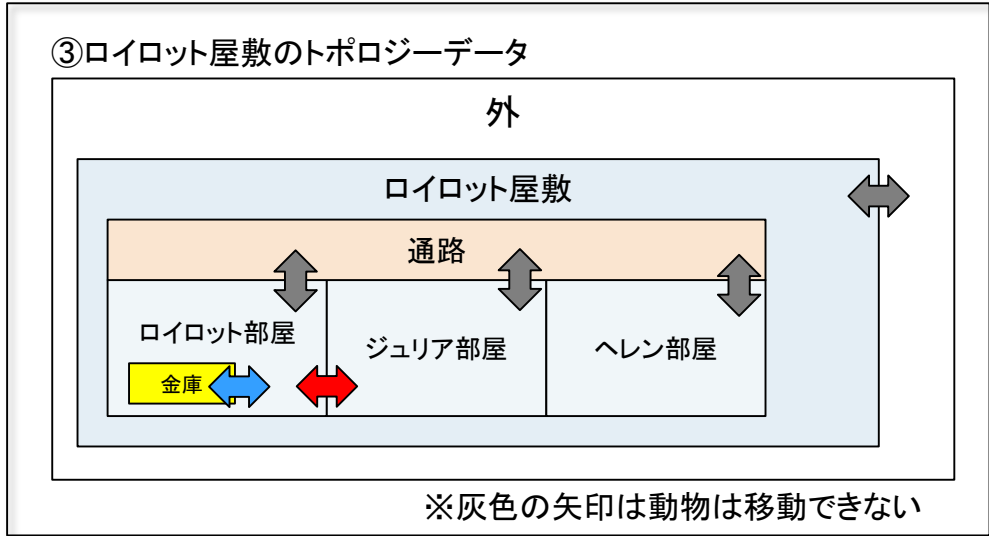
加えた外部知識

■ 主な外部知識は以下

- ①密室殺人の殺害方法リスト
- ②インドに生息する生き物とその属性、行動リスト→回収できる ⇒蛇、トカゲ、蜘蛛
- ③ロイロット屋敷のトポロジーデータ

①密室殺人の殺害方法リスト(自殺を除く、全てのケースを考慮)

- ・部屋外からの殺害 殺害は外で時間差
薬物
- ・室内で生きている被害者を外から殺害
隙間凶器
有毒生物
- ・自殺



②インドに生息する生き物		
	大きさ	毒性
牛	大	
象	大	
虎	大	
馬	大	
ロバ	大	
ラクダ	大	
ヤク	大	
ゾッキョ	大	
鹿	大	
山羊	大	
羊	大	
猿	中	
犬	中	
猫	中	
ウサギ	中	
鳥	中	
コウモリ	中	
鶴	大	
七面鳥	大	
カラス	中	
にわとり	中	
鳩	中	
魚	小	
カニ	小	
トカゲ	小	毒
ヒル	小	毒
蚊	小	
アリ	小	毒
クモ	小	毒
蛇	小	毒
ネズミ	小	
ハエ	小	

⑥Alloyを使って、述語論理の充足可能問題(SAT)を解く。 3つの場面を設定

■手順の目的

- 「犯行の可能性の列挙」を、ナレッジグラフ＋補完知識の述語論理式の充足可能問題(以下、SAT問題)ととらえる。
- SAT問題は、形式手法AlloyAnalyzerを使って解く。

■手順の概要

- ナレッジグラフと、外部知識を、述語論理で記述。
- 述語論理は、Alloy言語形式に変換する。
- AlloyAnalyzerは、Alloy言語をCNF(連言標準形)に変換し、SAT問題を解き、可能性の状況を列挙する。
- 複数の可能な状況(解)を、限定するため、“合理的な制約条件”を付けくわえ、1つの可能性に絞る。

ここで、「合理的な制約条件」とは、物語上に現れるナレッジグラフや、一般常識などの外部知識

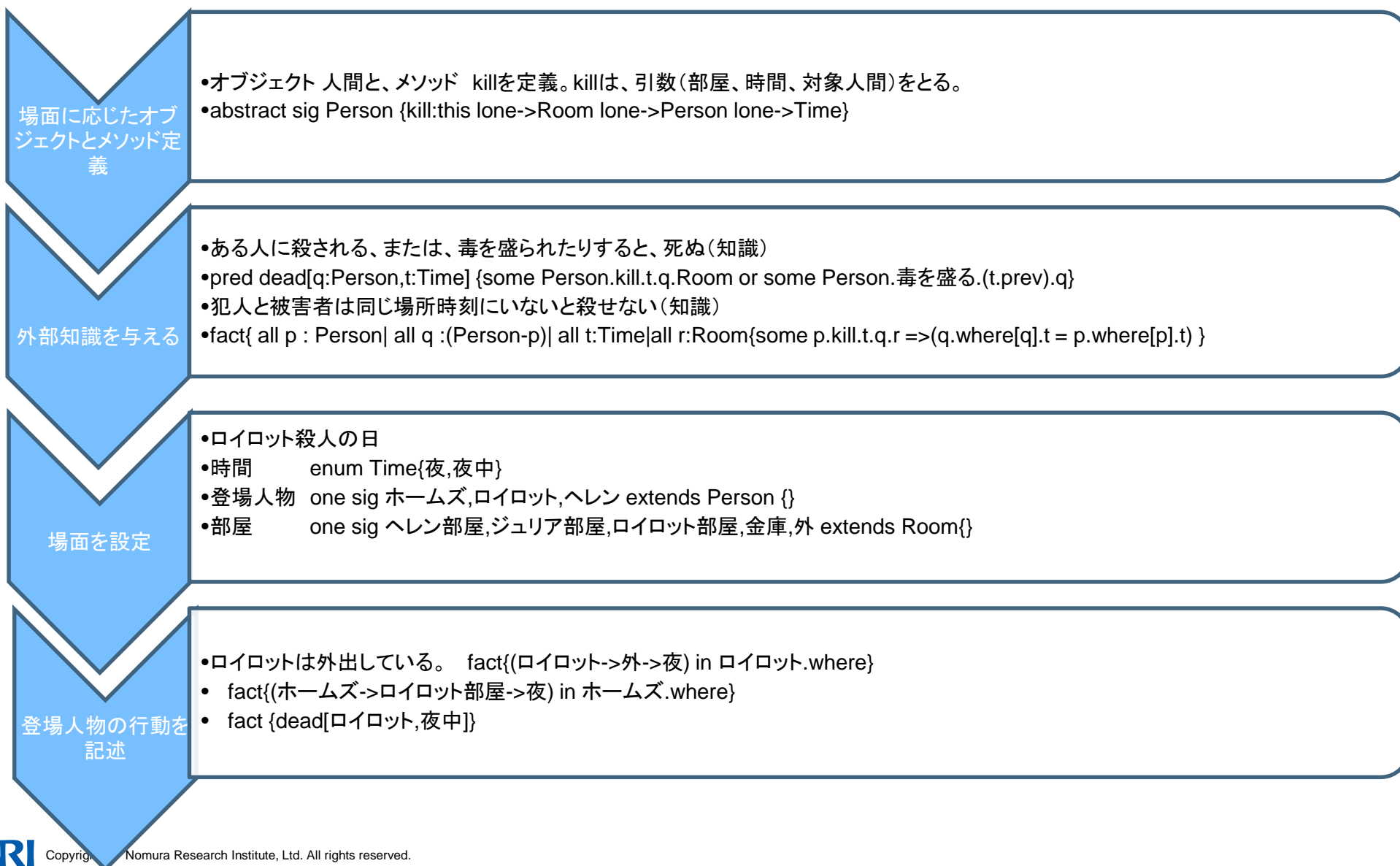
■設定した場面

場面1: ジュリア殺害の日

場面2: ヘレン殺害未遂の日

場面3: 殺人事件の真相の様相

⑥-1:形式手法Alloyの記述方法と知識の与え方



⑥-2:場面1と場面2でAlloyスクリプトに与えた設定・知識

■物語を2つの場面に分割する

●場面1と場面2の設定

	場面1 ジュリア殺害の日	場面2 ヘレン殺害未遂の日
時間	夜/深夜	夕方/夜/深夜
空間	ロイロット部屋、ジュリア部屋、ヘレン部屋、外、金庫	ロイロット部屋、ジュリア部屋、ヘレン部屋、外、金庫
登場人物	ヘレン、ジュリア、ロイロット	ヘレン、ロイロット、ホームズ
その他		殺害行為の失敗について追加

■2つの場面に共通部分と、各場面の状況を記述する部分について説明する。

●前者については、場面で異なる部分を赤字で注記。

⑥-3: 場面1と場面2 登場主体に関する記述

■ 登場主体に関する部分は、場面によって犯行阻止を表現するためのメソッドを持たせている。

- 添付したスクリプト本文で確認してほしい。

ジュリア殺害の日

```

////////////////////////////////
////
/////登場主体に関する記述
/////
////////////////////////////////
--登場する動物・人物はヒヒ,チーター,小さい有毒生物,ヘレン,ジュリア,ロイロット
abstract sig Character {
//登場主体は、ある時間に1つの場所にいる。
enter: this -> Room one -> Time,
}
//動物
abstract sig Animal extends Character {}
one sig ヒヒ,チーター,小さい有毒生物 extends Animal {}
//人間
abstract sig Person extends Character {
//人は、ある時間にある方法で自分以外を殺す意思を持つことがある
Kill: this lone -> lone ( Person - this ) -> Method -> Time,
//人は、ある時間にある部屋の施錠をすることがある
lock: this -> Room -> Time,
}
one sig ヘレン,ジュリア,ロイロット extends Person {}

```

ヘレン殺害未遂の日

```

////////////////////////////////
////
/////登場主体に関する記述
/////
////////////////////////////////
--登場する動物・人物はヒヒ,チーター,小さい有毒生物,ヘレン,ホームズ,ロイロット
abstract sig Character {
//登場主体は、ある時間に1つの場所にいる。
enter: this -> Room one -> Time,
}
//動物
abstract sig Animal extends Character {}
one sig ヒヒ,チーター,小さい有毒生物 extends Animal {}
//人間
abstract sig Person extends Character {
//人は、ある時間にある方法で自分以外を殺す意思を持つことがある
Kill: this lone -> lone ( Person - this ) -> Method -> Time,
//人は、ある時間にある部屋の施錠をすることがある
lock: this -> Room -> Time,
//人は、ある時間に誰かに反撃を加えることがある
Intercept: this -> lone Person -> Time
}
one sig ヘレン,ホームズ,ロイロット extends Person {}

```

⑥-3: 場面1 殺人方法に関する記述

■「密室殺人における殺害方法のリスト+直接(対面)の殺人」をもとに、全ての殺害方法の可能性を記述。

ジュリア殺害の日

```

////////////////////////////////
////殺人方法に関する記述
////////////////////////////////
//殺される人は、殺す意思を持たない
fact {
all p: Person | all t: Time { 殺害[Person,p,t] => no p.Kill}
}
--考えられる殺害方法は直接, 薬物, 隙間凶器, 小生物である。
abstract sig Method {
}
one sig 直接, 薬物, 隙間凶器, 小生物 extends Method {}
//殺害は、直接, 薬物, 隙間凶器, 小生物のいずれかで行われる。
pred 殺害[p : Person , q : Person , t:Time]{直接[p, q, t] || 薬物[p, q, t] || 隙間[p, q, t] || 生物[p, q, t]}

//直接は、pが直接殺害する意思を持っていて、pとqは時刻tに同じ部屋にいると行われる。
//薬物は、pが薬殺する意思を持っていて、pとqは殺害時刻以前t.prevsに同じ部屋にいると行われる
//隙間は、pが隙間凶器で殺害する意思を持っていて、pとqは時刻tに別の部屋にいるが、pの部屋とqの部屋は(隙間)行き来できると行われる。
//生物は、pが小生物で殺害する意思を持っていて、pとqは時刻tに別の部屋にいる。そして、ある動物aがいて、pとは別の部屋にいて、aが襲うと行われる。
//※襲うは、小生物で殺害する意思を持っている人pがいて、動物aがそれ以外の人qと同じ部屋にいる。

pred 直接[p : Person , q : Person , t:Time]{ one p.Kill.t.直接.q && no p.Kill.Time.(Method - 直接) && Together1[p, q, t]}
pred 薬物[p : Person , q : Person , t:Time]{ one p.Kill.t.薬物.q && no p.Kill.Time.(Method - 薬物) && Together1[p, q, (t.prevs)]}
pred 隙間[p : Person , q : Person , t:Time]{ one p.Kill.t.隙間凶器.q && no p.Kill.Time.(Method - 隙間凶器) && not Together1[p, q, t] && Passable[(~(p.enter.t)).p, (~(q.enter.t)).q]}
pred 襲う[a : Animal, t: Time ]{one p:Person | one p.Kill.t.小生物 && no p.Kill.Time.(Method - 小生物) && one q:( Person - p ) | Together1[a,q,t]}
pred 生物[p : Person , q : Person , t:Time]{ one p.Kill.t.小生物.q && no p.Kill.Time.(Method - 小生物) && not Together1[p,q, t] && one a:Animal | not Together1[p,a, t] && not Together2[p,a, t] && 襲う[a,t]}

--殺人方法による結果の定義
//凶器、隙間凶器使うと外傷ができる、遅効性薬物使うと薬物反応ができる
pred 外傷[p: Person ] { one t:Time | 直接[ Person, p, t ] || 隙間[ Person, p, t ] }
pred 毒物[p: Person ] { one t:Time | 薬物[ Person, p, t ] }
--殺人による結果:殺害された人は死ぬ。そうでない人は生きている。
pred Dead[p: Person, t: Time]{殺害[Person,p,t] }
pred Alive[p: Person, t: Time]{not Dead[p,t]}

```

⑥-3: 場面2 殺人方法に関する記述

■「密室殺人における殺害方法のリスト+直接(対面)の殺人」をもとに、全ての殺害方法の可能性を記述。

ヘレン殺害未遂の日

```

////////////////////
/////殺人方法に関する記述
////////////////////
//犯行の阻止が起きるということは、誰かが誰かの殺害を企図している。阻止をする人は、殺す意思を持たない。
fact {
  all p: Person | all q: Person | all t: Time {one p.Intercept.t.q => one q.Kill.t.Method.(Person - p) && no p.Kill }
}
--考えられる殺害方法は直接, 薬物, 隙間凶器, 小生物である。
abstract sig Method {
}
one sig 直接, 薬物, 隙間凶器, 小生物 extends Method {}
//殺害は、直接, 薬物, 隙間凶器, 小生物のいずれかが、失敗しなかったときに行われる。
pred 殺害[p: Person, q: Person, t: Time]{ 直接[p, q, t] || 薬物[p, q, t] || 隙間[p, q, t] || 生物[p, q, t] && not 失敗[p, t] }

//直接は、pが直接殺害する意思を持っていて、pとqは時刻tに同じ部屋にいると行われる。
//薬物は、pが薬殺する意思を持っていて、pとqは殺害時刻以前t.prevsに同じ部屋にいると行われる
//隙間は、pが隙間凶器で殺害する意思を持っていて、pとqは時刻tに別の部屋にいるが、pの部屋とqの部屋は(隙間)行き来できると行われる。
//生物は、pが小生物で殺害する意思を持っていて、pとqは時刻tに別の部屋にいる。そして、ある動物aがいて、pとは別の部屋にいて、aが襲うと行われる。
//※襲うは、小生物で殺害する意思を持っている人pがいて、動物aがそれ以外の人qと同じ部屋にいる。

pred 直接[p: Person, q: Person, t: Time]{ one p.Kill.t.直接.q && no p.Kill.Time.(Method - 直接) && Together1[p, q, t] }
pred 薬物[p: Person, q: Person, t: Time]{ one p.Kill.t.薬物.q && no p.Kill.Time.(Method - 薬物) && Together1[p, q, (t.prevs)] }
pred 隙間[p: Person, q: Person, t: Time]{ one p.Kill.t.隙間凶器.q && no p.Kill.Time.(Method - 隙間凶器) && not Together1[p, q, t] && Passable[(~(p.enter.t)).p, (~(q.enter.t)).q] }
pred 襲う[a: Animal, t: Time]{ one p: Person | one p.Kill.t.小生物 && no p.Kill.Time.(Method - 小生物) && one q: (Person - p) | Together1[a, q, t] }
pred 生物[p: Person, q: Person, t: Time]{ one p.Kill.t.小生物.q && no p.Kill.Time.(Method - 小生物) && not Together1[p, q, t] && one a: Animal | not Together1[p, a, t] && not Together2[p, a, t] && 襲う[a, t] }

//誰かが殺す意思を持つも、それが阻止されたときは失敗となる。
pred 失敗[p: Person, t: Time]{ one p.Kill.t.Method && one Person.Intercept.t.p }
--殺人方法による結果の定義
//凶器、隙間凶器使うと外傷ができる、遅効性薬物使うと薬物反応ができる
pred 外傷[p: Person]{ one t: Time | 直接[Person, p, t] || 隙間[Person, p, t] || ( one p.Kill.t.(隙間凶器 + 直接) && 失敗[p, t] ) }
pred 毒物[p: Person]{ one t: Time | 薬物[Person, p, t] || ( one p.Kill.t.薬物 && 失敗[p, t] ) }
--殺人による結果
pred Dead[p: Person, t: Time]{ 殺害[Person, p, t] || 失敗[p, t] }
pred Alive[p: Person, t: Time]{ not Dead[p, t] }

```

⑥-3: 場面1と場面2 建物に関する記述

■ 場面に共通する部分は、登場主体・殺人方法・建物/所在を記述した。

- 2つの場面で若干異なる部分もある。スクリプト本文を確認してほしい。

```

//////////
//////建物に関する記述
//////////
--建物に関する定義
--登場する建物はヘレン部屋,ジュリア部屋,ロイロット部屋,金庫,外である。
abstract sig Room {
    owner : set Person,
    pass: set Room
}
one sig ヘレン部屋,ジュリア部屋,ロイロット部屋,金庫,外 extends Room{}
fact {
//部屋はownerが内側からしかlockできない。金庫はownerがlockできる
    all r: ( Room - 金庫 ) | all p: Person | all t: Time { some p.lock.t.r => p in r.owner && some p.enter.t.r}
    all r: 金庫 | all p: Person | all t: Time { some p.lock.t.r => p in r.owner}
//部屋が、ある人にlockされてたら、その人以外は入れない。
    all disj p, q: Person | all r: (Room - 金庫) | all t: Time { some p.lock.t.r => no q.enter.t.r }
//金庫に人は入れない
    all p: Person {not p -> 金庫 in p.enter.Time}
//金庫はロイロット不在のときは施錠されている。入っているものはロイロットが鍵をかけている限り出れない。
    all t: Time {no ロイロット.enter.t.ロイロット部屋 => some ロイロット.lock.t.金庫 }
    all t: Time {one ロイロット.lock.t.金庫 => lone Character.enter.t.金庫 }
//隙間がある部屋間
    ロイロット部屋.pass=(ジュリア部屋 + ロイロット部屋)
    ジュリア部屋.pass=(ジュリア部屋 + ロイロット部屋)
    all r:( Room - ロイロット部屋 - ジュリア部屋) {no r.pass}
}

```


⑥-3: 場面1と場面2 所在に関する記述

■ 場面に共通する部分は、登場主体・殺人方法・建物/所在を記述した。

- 2つの場面で若干異なる部分もある。スクリプト本文を確認してほしい。

```

//////////
/////所在に関する記述
//////////
--所在に関する定義
--同部屋にいることの定義
pred Together1[ p : Character , q : Character , t:Time ]{ ~(p.enter.t).p = ~(q.enter.t).q}
pred Together2[ p : Character , q : Character , t:Time ]{ ~(p.enter.t).p.owner = ~(q.enter.t).q.owner}
pred Passable[r:Room, s:Room]{ s in r.pass}
pred Alone[ p : Character , t:Time ]{ not Together1[p, Person,t] }

--動物が動き回れる範囲
fact{
//動物は外と屋内を行き来できない
all a:Animal { a -> 外 in a.enter.夜 => a -> 外 in a.enter.深夜}
all a:Animal { not a -> 外 in a.enter.夜 => not a -> 外 in a.enter.深夜}
all a:Animal {
    lone a.enter.Time.金庫
    lone a.enter.Time.ジュリア部屋
    lone a.enter.Time.ロイロット部屋
    lone a.enter.Time.外
    all r:( Room - 金庫 - ジュリア部屋 - ロイロット部屋 - 外 ) { no a.enter.Time.r}
}
}

```

⑥-4: 場面1:「ジュリア殺害の日」とAlloyによる結果

■ 場面に共通する部分は、登場主体・殺人方法・建物/所在を記述した。

- 2つの場面で若干異なる部分もある。スクリプト本文を確認してほしい。

```
//////////
//////場面に関するFACT
//////////
```

```
//場面に関する情報
//場所に関する情報(言及あり)
fact {
ジュリア部屋.owner=ジュリア           //
ヘレン部屋.owner=ヘレン                 //
ロイロット部屋.owner=ロイロット         //
金庫.owner=ロイロット                   //
ジュリア.lock.深夜=ジュリア -> ジュリア部屋 //
ヘレン.lock.深夜=ヘレン -> ヘレン部屋     //
Together1[ヘレン, ジュリア, 夜]         //
ヘレン.enter.夜=ヘレン -> ヘレン部屋     //
ジュリア.enter.夜=ジュリア -> ヘレン部屋 //
ヘレン.enter.深夜=ヘレン -> ヘレン部屋   //
ジュリア.enter.深夜=ジュリア -> ジュリア部屋 //
ロイロット.enter.深夜=ロイロット -> ロイロット部屋 //
ヒヒ.enter.夜=ヒヒ -> 外                 //
チーター.enter.夜=チーター -> 外         //
}
//状態に関する情報(言及あり)
fact {
Dead[ジュリア,深夜]                     //
Alive[ヘレン,夜]                         //
Alive[ジュリア,夜]                       //
Alive[ヘレン,深夜]                       //
Alive[ロイロット,夜]                     //
Alive[ロイロット,深夜]                   //
not 外傷[ジュリア]                       //
not 毒物[ジュリア]                       //
}
```

モデルを限定するための付加条件

```
//ヘレンは善意の正直者。
pred honest[p: Person]{ no p.Kill}
fact {
//ヘレンは殺す意思を持たない
    honest[ヘレン]    //
}

//これより下、結果に影響がない範囲で与えた計算上都合上の束縛条件
//場面からの推測(外部知識でもある)
fact {
//ヘレンとジュリアが夜にヘレン部屋にいる間
//ヘレン部屋が施錠されているか定かではない事実から
//ヘレン部屋に人物が入り出る解が出てきてしまう。
ロイロット.enter.夜=ロイロット -> ロイロット部屋
no Character.enter.夜.ジュリア部屋

//外は誰のものでもない
//ownerを付与すると、施錠するかどうかの解が増えてしまうため
no 外.owner
}
```

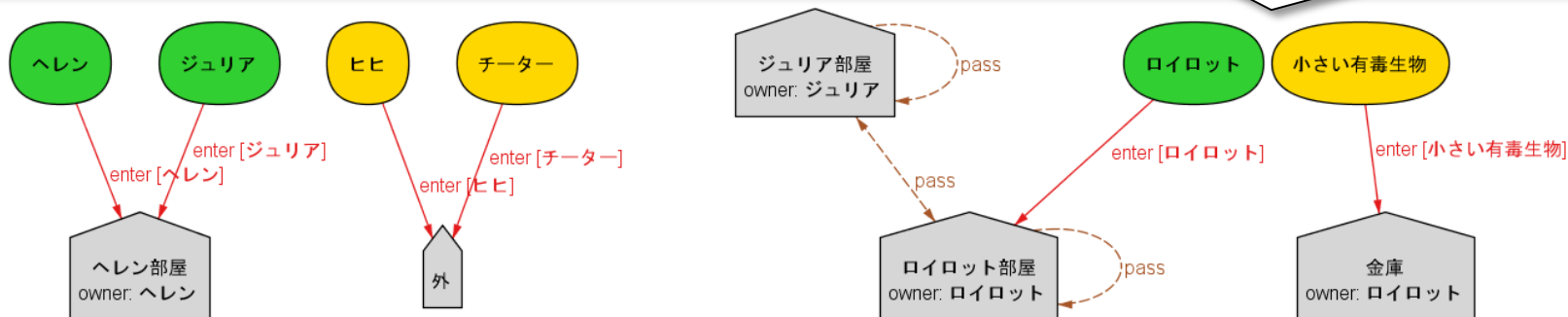
以上のような一階述語論理と事実からこれを満たす解を探す

⑥-4:場面1:「ジュリア殺害の日」とAlloyによる結果(青字)

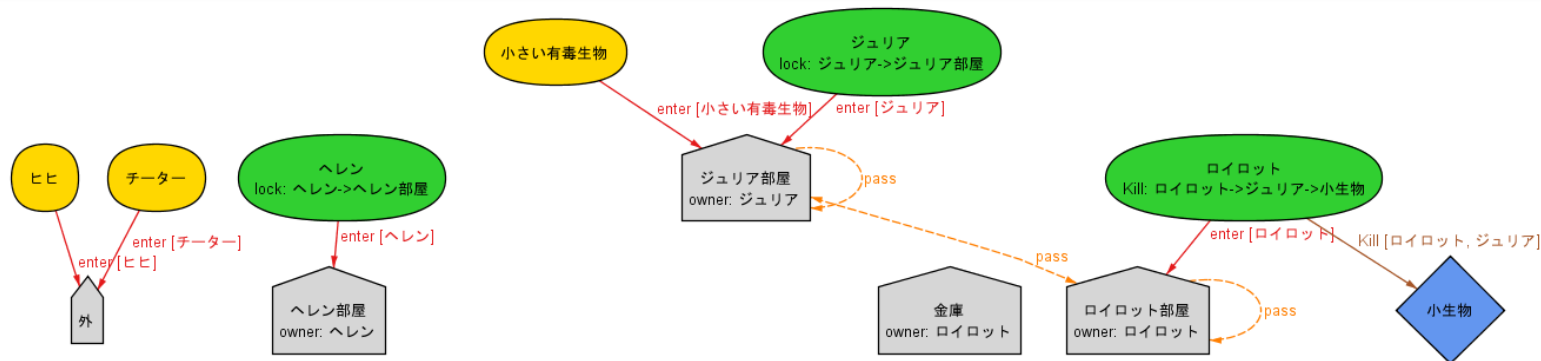
■ Royslottの犯行の可能性のみが残り、当時の人物・動物・モノの所在や状態の可能性が解として提示される

夕方において、以下の2パターンの可能性が見られた。
ロイロットがロイロット部屋にいる ⇒小さい有毒生物が金庫かロイロット部屋にいる

夜



深夜



ロイロットが、ジュリアに対し、小生物での殺害の意思を持ち、実行して、ジュリアを殺害。

⑥-5: 場面2:「ヘレン殺害未遂の日」とAlloyによる結果

■ヘレン殺害未遂の日では、殺害の方法に関して修正を加えた。

- 殺害時に、邪魔・反撃が入ると失敗し、自分が死ぬ

```

////////////////////
//////場面に関するFACT
////////////////////
//場所に関する情報(言及あり)
fact {

//共通部分
金庫.owner=ロイロット
no 外.owner
ロイロット部屋.owner=ロイロット

//場面に拠る状況
ヘレン部屋.owner = (ヘレン + ロイロット)
ジュリア部屋.owner=(ヘレン + ホームズ)
}
//状態に関する情報(言及あり)
fact {
//夕方
小さい有毒生物.enter.夕方!=小さい有毒生物 -> 外
小さい有毒生物.enter.夕方!=小さい有毒生物 -> ジュリア部屋
小さい有毒生物.enter.夕方!=小さい有毒生物 -> ロイロット部屋
小さい有毒生物.enter.夕方!=小さい有毒生物 -> ヘレン部屋
ロイロット.lock.夕方=ロイロット -> 金庫
ロイロット.enter.夕方=ロイロット -> 外

//夜
小さい有毒生物.enter.夜!=小さい有毒生物 -> ジュリア部屋
one ロイロット.lock.夜.金庫
ホームズ.enter.夜=ホームズ-> 外
ヘレン.enter.夜=ヘレン-> ジュリア部屋
ヘレン.lock.夜=ヘレン-> ジュリア部屋
ヒビ.enter.夜=ヒビ -> 外
チーター.enter.夜=チーター -> 外

```

```

//深夜
one ホームズ.Intercept.深夜
ロイロット.enter.深夜=ロイロット -> ロイロット部屋
ホームズ.enter.深夜=ホームズ-> ジュリア部屋
ホームズ.lock.深夜=ホームズ-> ジュリア部屋
ヘレン.enter.深夜=ヘレン-> ヘレン部屋

```

Together1[ヘレン,ホームズ,夕方]

```

Alone[ロイロット,夜]
Dead[ロイロット,深夜]
Alive[ロイロット,夜]
Alive[ロイロット,夕方]

```

```

Alone[ヘレン,夜]
Alive[ヘレン,深夜]
Alive[ヘレン,夜]
Alive[ヘレン,夕方]

```

```

Alive[ホームズ,深夜]
Alive[ホームズ,夜]
Alive[ホームズ,夕方]
}

```

```

//解の個数を狭めるための束縛条件(推理に本質でない)
//束縛のための付与条件
pred honest[p: Person]{ no p.Kill}
fact {
no ヘレン.enter.夕方.外
ヘレン.enter.夕方=ヘレン-> ヘレン部屋
not 外傷[ロイロット]
not 毒物[ロイロット]

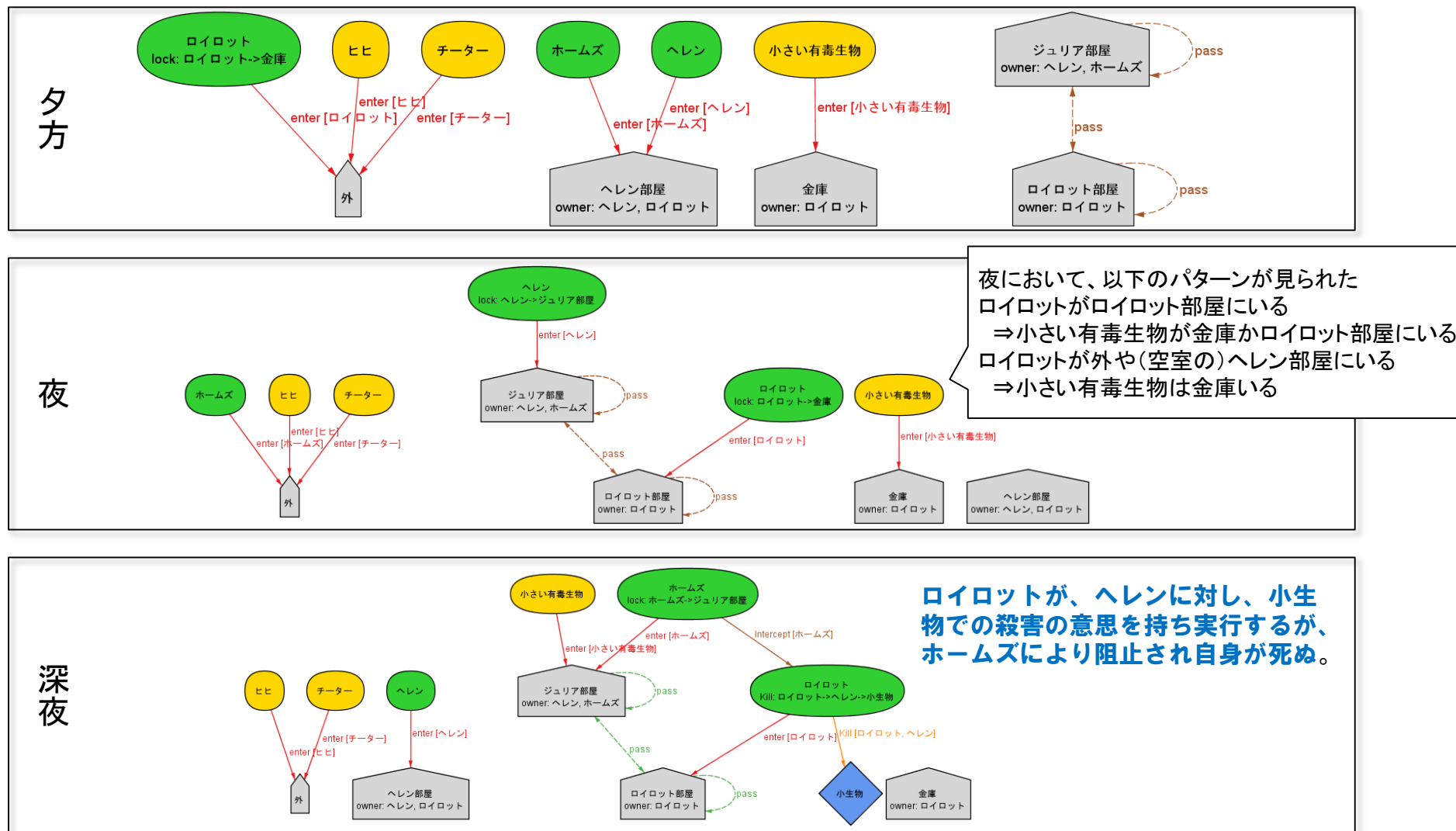
//ヘレンはhonestである。
honest[ホームズ]
honest[ヘレン]
}
//扉にlockがかかるかどうかで解のパターンが増えるので限定
pred show {
#Person.lock <5
//one ロイロット.Kill.Time.Method.ホームズ
}
run show

```

以上のような一階述語論理と事実からこれを満たす解を探す

⑥-5:場面2:「ヘレン殺害未遂の日」とAlloyによる結果(青字)

■ロイロットが犯行を試み、失敗する解が提示される



⑥-5:場面3:「事件の真相の様相」とAlloyによる結果 ～コナンドイルの物語の階層構造～

■ ヘレンの証言のみで物語が進む部分について、論理と可能性から別の結論を考える。

物語の階層構造を考慮する理由

①ヘレンが死んだ後に、ワトソン(視点人物)がこの話を語りだす。

②前半は、ヘレンの発言となっている。

＝ヘレンの語りをベースに、ホームズは推理する。

＝ヘレンの発言(証言)を嘘とすると、後半のホームズは間違う。

③ヘレンの怪しい点

- ・ヘレンにも動機あり。
- ・ロイロットをホームズ屋敷まで追いかけさせ、ロイロット部屋を留守に！

場面3の設定のポイント

「ヘレンがジュリア殺害の真相」を知っていたか、知らなかったかを考慮し、**様相論理**で推論を行う。

- ・ヘレン発言をすべての嘘にすると、FACTが足らずに解が出ない
- ・ヘレン発言毎に真偽値を振り分ける方法は、煩雑で今回はできず。

参考：https://www.nri.com/jp/knowledge/publication/fis/kinyu_itf/1st/2018/10/08



⑥-5: 場面3: 「事件の真相の様相」とAlloyによる結果 Alloyの様相論理公理設定

```

sig World {ジュリア殺害真相: Hat} //世界はいくつあってもよい。(oneを入れていない)
abstract sig Hat {}
one sig ヘレン知る, ヘレン知らない extends Hat {} -- ジュリア殺害真相をヘレンが知るか知らないか、世界はどちらか。
abstract sig Honest {}
one sig 嘘つき, 正直 extends Honest {}
one sig RealWorld extends World {}
abstract sig Person {access: World -> World, Honest: Honest} //人物は世界にアクセスできる、かつ正直か嘘つきかの属性を持つ。
one sig ホームズ extends Person {} //登場人物はホームズとヘレンのみ
one sig ヘレン extends Person {}
abstract sig liveordead {}
one sig ロイロット死 extends liveordead {} //ロイロットは死か

```

```

// 様相論理の公理 -- 世界の同値関係と推移関係の定義
fact S5 {all p : Person | {
    // reflexive
    all w : World | all w': World | w' in p.access[w] => w in
    p.access[w]

    // symmetric and transitive (if u->v and u->w then v->w)
    all w : World | all w' : p.access[w] | all w'' : p.access[w] | w''
    in p.access[w']
}}

//すべての世界
fact DifferentWorldsAreDifferent {
    all w : World | all w' : World {(w.ジュリア殺害真相 = w'.ジュリア殺害真相) => w
    = w'}
}

//世界は、ジュリア殺害方法をヘレンが知るか、ヘレンが知らないかの2つの世界がある。
fact {
    all w: World {(w.ジュリア殺害真相 = ヘレン知る)
    or (w.ジュリア殺害真相 = ヘレン知らない) }}

```

⑥-5: 場面3: 「事件の真相の様相」とAlloyによる結果 Alloyの様相論理命題の外部知識の設定

```

// ヘレン自身は、世界がどちらか知っている。
fact {all w: World | ヘレン.access[w] = {w': World | w.ジュリア殺害真相 = w'.ジュリア殺害真相}}
// ヘレンが正直ならば、ホームズはヘレンが知っているか知らないかを知っている。
fact { (ヘレン.Honest=正直) => all w: World | ヘレン.access[w] = ホームズ.access[w]}
//ヘレンが嘘つきならば、ホームズは情報がない。
fact { (ヘレン.Honest=嘘つき) => no ホームズ.access}

// 人が世界を知っていることを定義している。
pred knows[p: Person, w: World, u: set World] {
    all w1: (p.access[w] & u) | all w2: (p.access[w] & u) | w1.ジュリア殺害真相 =
w2.ジュリア殺害真相}
//ロイロットは死んだことをFACTとして記述。
fact{ロイロット死= liveordead}
//ホームズは正直としている。
fact{ホームズ.Honest=正直}

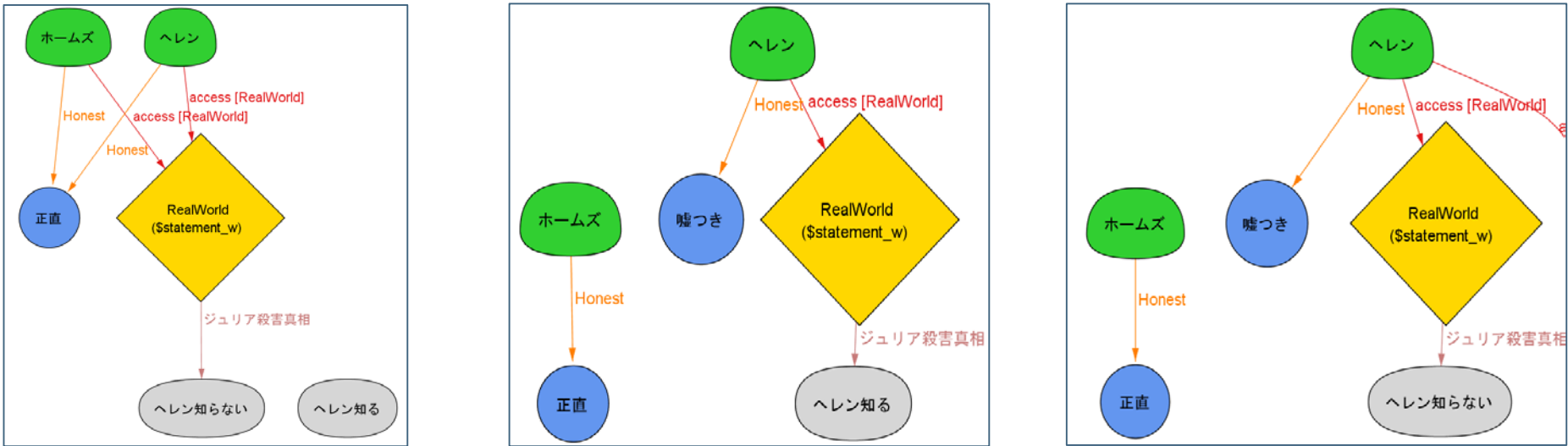
pred statement[w:RealWorld] {
    //ヘレンが殺害方法を知っており、かつ、ホームズが、ジュリア殺害方法をヘレンが知っていること
    を知っていたならば、ロイロットは死なない。
    ( w.ジュリア殺害真相 = ヘレン知る and ヘレン.access[w] in ホームズ.access[w] ) => ( ロイロット
死 != liveordead ) }
//ヘレンが、アクセスできる世界をホームズが知らないならば、ジュリアは殺害の真相をしっている。
assert ヘレンのたぐらみ{all w': ヘレン.access[World] | not knows[ホームズ,w',World] => w'.ジュリ
ア殺害真相 = ヘレン知る}
//check ヘレンのたぐらみ
run statement

```

様相論理命題の外部知識

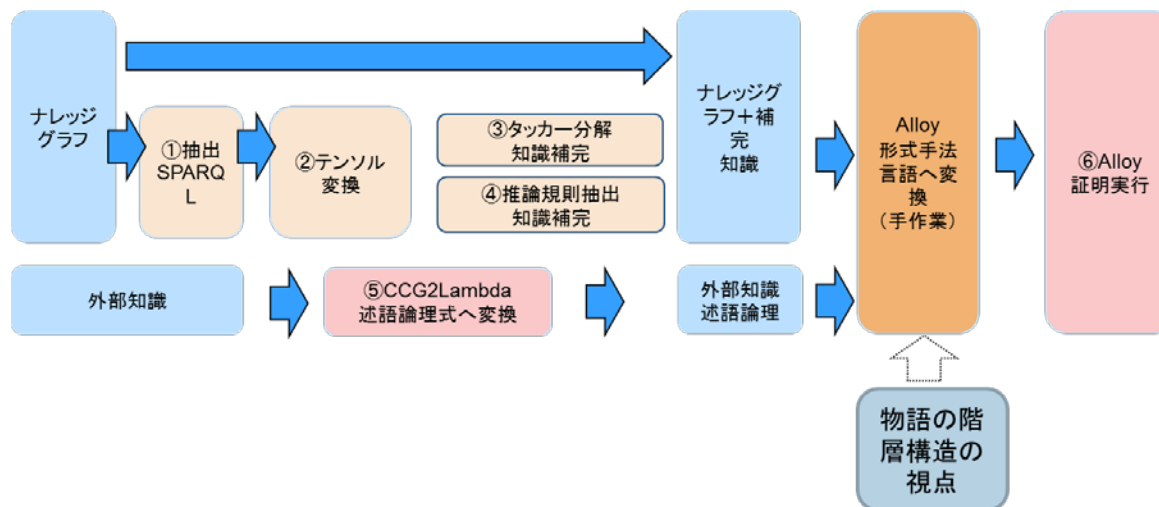
- ①ヘレンが殺害真相を知っており、かつ、ジュリア殺害真相をヘレンが知っていることをホームズが、知っていたならば、ロイロットは死なない。
- ②ヘレンが正直ならば、その時に限り、ヘレン殺害真相を知っていたか、知らなかったか、ホームズは知る。

⑥-5:場面3:「事件の真相の様相」とAlloyによる結果



ヘレン		ホームズ	Alloy結果	
ジュリアの状態A	ジュリアの意思	ジュリアの状態がAであることを	結果	考察
ジュリア殺害真相を知っていた	正直に話す	知る	×	ホームズは、ヘレンから真相を伝えられ、ロイロットは死なない。
		知らない	×	ホームズの能力から、起きない。
	うそを言う	知る	×	ヘレンが嘘つきのため、起きない。
		知らない	○	ヘレンの策略のため、ロイロットは死ぬ。
ジュリア殺害真相を知らなかった	正直に話す	知る	○	物語のメイン解釈。ホームズの落ち度で最後の発言の根拠
		知らない	×	ホームズの能力から、起きない。
	うそを言う	知る	×	ヘレンが嘘つきのため、起きない。
		知らない	○	事故は偶然起きた。

C)まとめ1 アプローチ全体と各プロセス



- 「犯行状況の可能性列挙」を、述語論理式の充足可能問題(SAT問題)にすることで、推論過程でどのファクトを使うかの取捨選択自体を、プログラムに任せることができた。
- SAT問題により、後ろ向き推論では出てこない、真犯人の可能性を上げることができた。
- テンソル分析における知識補完は、タッカー分解手法ではうまくいった。機械的な知識補完は、人がその知識を使うかを考慮しなければならないが、人が気づかないような示唆を与えることが可能。
- Alloyは、状況やFACTの設定が容易で、かつ述語論理式をSAT問題におとすための非常に強力なツールとなる。

C)まとめ2 技術的知見と課題

■ テンソル形式での知識補完

Tucker分解

- 大規模疎テンソルの補完は、補完精度などが議論されていて、適用してよい手法かは再考の余地がある
- Tucker分解の分解一意性を与えるような、制約条件を見つけることが必要かもしれない。
- 大規模疎テンソルに対しての線形代数計算に耐えうる、強力な計算資源が必要と感じた。
 - ・ 現実的な計算を考えると、テンソルを人物に絞ってしまい、残りの部分は外部知識となってしまった。
 - ・ 当該チャレンジ内容とは別に、内部知識でどこまで知識を補完できるかについて、非常に興味を持った。

推論規則抽出

- 事実関係の列挙のケースとして紹介されている当該方法について、時間関係がある今回のケースでの利用は難しいと感じた。
- 推論規則抽出においては、シーンの区切り、ごく少数カウントされるケースを省くなどの前提知識を設けることで現実的な抽出ができた。
- テンソルのベキ乗という形式でなく、幅優先探索を利用して高速化したことは大きい。

■ 機械学習的なパラメータは外部知識の一部として、記述するべきと考えた。

■ AlloyでSAT問題を解くための時間がかかるために、モデルを限定する必要があった。

- 与えた束縛条件における解の全数提示には、大規模な並列計算が必要となる。
- そのために、外部知識(制約条件)を与えて、モデルを限定する必要があった。

■ 本件では、行列計算やSAT問題がメインとなり、大規模な並列計算環境や量子コンピュータのような計算環境があれば、全数・網羅性を持たせた解析ができた。

C)まとめ3 物語の真相

- ロイロットの犯行であることは、解として提示される。
- ロイロット殺害に関して、本文中のfactだけだとヘレン犯人の可能性がSAT問題の解として提示される。
- 以上から、場面3を考える必要があった。後ろ向き推論では、この可能性に気づけないので、非常に意味のある結果とを感じる。

※ AlloyAnalyzerのスクリプトでHonest[ヘレン]をコメントアウトすることで、以下の解の候補が提示される。

- 解の候補

- ・ ヘレンがロイロットと共謀して、ジュリアを殺害した可能性
 - ・ ヘレンがロイロットの手法を知り、ホームズを利用してロイロットを殺害した可能性もある

- 特に、ヘレンが犯人である可能性を示唆する文面も多い

- 動機

- ・ ロイロットが近隣に迷惑をかけ、身内として迷惑している。
 - ・ ロイロットから暴力を受けている。
 - ・ ヘレンもロイロットとの遺産相続の利害関係者である。

- 示唆する点

- ・ ジュリア殺害の状況がヘレンの証言でしか確かめられていない。
 - ・ “蛇が必ず相手を咬む保証はない。もしかすると、一週間位、彼女は毎夜その毒牙から逃れていたかもしれない。”という発言から、ヘレンは偶然犯行の方法を知ったかもしれない。

- 以上の可能性は、文章中での事実だけでは否定することができない。
- ヘレンが一部事件に関わった可能性は、シャーロックホームズを研究しているシャーロキアンのなかでも注目されておらず、本チャレンジで機械的に抽出された結果として、非常に興味深い。

3. 実行プログラム

■ 自作したプログラムと利用した外部ライブラリー

- スクリプトは応募時のファイルに添付した。
- 全て無償のソフトウェアである。

■ 自作したプログラム

- 英文→RDF(精度が悪く、今回未利用)
- 3つ組抽出SPARQLクエリー : (資料中に記載@SPARQL)
- RDF(ntファイル)→テンソル(mtxファイル) : (簡易なシェルスクリプトのため略)
- テンソル タッカー分解 : Tucker4SH.py (資料中に記載@Python)
- テンソル解析 推論規則 抽出 : DFS4SH.c (添付@C言語)
- Alloy用モデル 3場面(ジュリア殺害、ロイロット殺害、事件の真相の様相) : (添付&資料中に記載@AlloyAnalyzer)

利用した外部ライブラリ

■ CCG2Lambda

- <https://github.com/mynlp/ccg2lambda>

■ Alloy Analyzer (alloy4.2_2015-02-22.jar)

- <http://alloytools.org/>

計算環境

■ Win7 Professional : 8GB

■ AWS t2xlarge: インスタンス 16GB 4コアCPU

NRI

未来創発

Dream up the future.