3B1-02

実時間統合シミュレーションのためのデータセットバージョン管理

Formalization of Version Control of Data-Set for Real-Time Integrated Simulation

野田 五十樹
NODA, Itsuki

(独) 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

A mathematical formalization of *version control* is proposed, in which manages consistency among data used in integrated simulations. Keeping consistency of data in integrated simulations for complex social phenomena like disaster and rescue is an important issue to validate these results, because multiple and delayed information are reported to the database. The proposed formalization gives fundamental framework of consistency among data simulation. I also investigate and discuss about cost of the management in several implementation style.

1. はじめに

災害など複雑な現象を計算機で扱う手法として、統合シミュレーションの構築が進められている [Tad03b, Tad03a]。これら統合シミュレーションの特徴は、関連ある事象の変化を各々予測するシミュレータと結合し、相互に計算結果を反映しながら全体としての現象をシミュレーションする点である。

このようなシミュレーションを具体的な問題に提供する場合、誤差を含む膨大な量のデータを扱い、それらの間の依存関係を適切に管理する必要がある。また、災害が実際に発生した直後に被害想定や救助計画立案のためにシミュレーションを用いる場合、災害現場のセンサーデータの報告に遅延が生じることを考慮する必要がある。このような条件をふまえた上で、シミュレーションに対して整合性のある初期データを提供する枠組みが必要となる。本報告では、このようなデータの整合性を形式的に定義し、統合シミュレーションへの入出力を担うデータベースに求められる機能を整理する。

2. データ管理に求められる要件

現場におけるシミュレーションの利用を考える場合、センサーから得られるデータがかならずしも真ではないことを前提としなければならない。特に同じ事象を観測する複数のセンサーがあった場合、それらが異なるデータを報告することは普通に起こり得る。また、データは必ずしもリアルタイムで入ってくるとは限らず、場合によってはシミュレーション開始後に入ってくる場合もある。一方、現場で利用されるシミュレーションではそのような遅延のある報告をいちいち待っていられない事が多い。

さらに、災害救助シミュレーション [Tad03b, Tad03a] 等の統合シミュレーションでは、膨大なデータに対し、複数のサブシミュレータが相互に依存して計算を行う。このため、整合性チェックは軽量でスケーラブルな枠組みである必要がある。

上記の要件をまとめると、統合シミュレーションのためのデータベースに求められる機能は次のようになる。

- ある事象の観測・推定として、複数あり得る値のどれをシミュレーションの初期値として用いたかを管理する手法。
- 複数のシミュレーション結果を統合する際に、そのデータの整合性をチェックできる機能。
- 遅延して報告された値に対し、報告以前に行われたシミュレーションとの整合性をチェックする機能。

A: 野田 五十樹、産業技術総合研究所、〒 135-0064 東京都青海 2-41-6, Tel: 03-3599-8230, Fax: 03-5530-2067

3. 形式的定義と定理

3.1 データベース

ここでは**データベース**を、**データ** の集まりとみなす。また **データ** とはある時刻のある **事物** について観測あるいは推定された個々の値とする。データベースは動的であるとする。すなわち、データベースへはシミュレーションなどとは非同期にデータが書き加えられるものとする。

以下に形式的定義を述べる。

事物 θ は、個々の物体やその属性、現象などの同定子である。例えば、“建物 X”、“車 Y の速度”、“地点 Z の震度” などは事物である。また、**事象** e は、事物 θ と時刻 t の対である。よって、事象 e は $\langle\theta, t\rangle$ とも記述される。例えば、“時刻 t における人 X の負傷度” などは事物 である。

データ $d_{\theta,t,s}$ は、**センサー**あるいは**シミュレーション** s によって観測・推定された事象 $\langle\theta, t\rangle$ の値である。観測や推定にはノイズや前提条件の違いがあるので、同じ事象に対し異なるデータがあり得る。s はセンサーの同定子あるいはシミュレーションのセッションに対応する**バージョン** (次節参照) に対応する。また、データがデータベースに報告された時刻を明記する場合は、$d_{\theta,t,s;\tau}$ ような記述を用いる。ただし、τ はデータベースにこのデータが報告された時刻である。

データベース $\mathbf{DB} = \{d_{\theta,t,s;\tau}\}$ は (将来に渡って報告されるものを含む) 全データの集合である。また、$\mathbf{DB}_{(0,\mathcal{T})} = \{d_{\theta,t,s;\tau} \in \mathbf{DB} | \tau < \mathcal{T}\}$ は、ある時刻 $\mathcal{T}$ におけるデータベースのスナップショット、すなわち、ある時刻 $\mathcal{T}$ までに報告されたデータの集合を表す。

3.2 バージョン

シミュレーション とは、データベース **DB** と以下のように入出力を行うものとして定義する。

1. ある事象の集合 $\boldsymbol{E}_{\text{ground}} = \{e_i | i\}$ に関するデータの集合 $\boldsymbol{D}_{\text{ground}}$ を、ある時刻 $\mathcal{T}_{\text{begin}}$ (一般にシミュレーション開始時刻) にデータベース **DB** から取得する。
2. 別の事象の集合 $\boldsymbol{E}_{\text{target}} = \{e_i | i\}$ についてのデータ $\boldsymbol{D}_{\text{target}}$ を推定する。
3. 推定されたデータ $\boldsymbol{D}_{\text{target}}$ を時刻 $\mathcal{T}_{\text{end}}$ (一般にシミュレーションの完了時刻) にデータベース **DB** に書き込む。

このシミュレーションの入出力に用いられるデータの集合 $(\boldsymbol{D}_{\text{ground}}, \boldsymbol{D}_{\text{target}})$ につけられた名前を **バージョン**と呼ぶ。よって、シミュレーションはあるバージョンのデータ集合をデータベースから受け取り、別のバージョンのデータ集合を作成してデータベースへ登録する処理とみなされる。

バージョン v は $\langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \boldsymbol{G} \rangle$ という対で定義される。ここで、 $\boldsymbol{E} = \{e_i | i\}$ は事象の集合、$\boldsymbol{D}$ はそのバージョンに属するデータ集合、$\mathcal{T}$ はデータベース **DB** を操作した時刻、$\boldsymbol{G} = \{u_i | i\}$ は $\boldsymbol{D}$ が直接依存するバージョンの集合を表す。これらの対の要素は以下の条件を満たさなければならない。

$$\boldsymbol{D} \subset \mathbf{DB}_{(0,\mathcal{T})}/E = \{d_{\theta,t,s;\tau} | \langle \theta, t \rangle \in \boldsymbol{E}, \tau < \mathcal{T}\}$$

3.3 バージョンの生成と根拠

新しいバージョンが生成されるには、**発生**、**抽出**、**算出**、**合同**の 4 種類の方法がある。

DB からセンサーにより直接観測されたデータのみを集めて新しいバージョン v をつくる場合、そのバージョンを **発生**するという。発生させられたバージョンは $v = \langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \phi \rangle$ のような対で表される。ここで、$\boldsymbol{E}$ は関連する事象、$\boldsymbol{D}$ はセンサーから直接得られたデータからなる集合、$\mathcal{T}$ は $\boldsymbol{D}$ を **DB** から集めた時刻である。

既存のバージョン $u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$ からいくつかのデータを取捨選択してバージョン v を作成する場合、そのバージョンを**抽出**するという。抽出された v は $\langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \{u\} \rangle$ という対で表される。ただし、$\mathcal{T}_v$ は抽出の時刻であり、また、$\boldsymbol{D}_v \subseteq \boldsymbol{D}_u$ および $\boldsymbol{E}_v = \{e | d_{\theta,t,s} \in \boldsymbol{D}_v, e = \langle \theta, t \rangle\}$ が満たされるものとする。

あるバージョン u を入力としてシミュレーションした結果をバージョン v として得る場合、そのバージョンを **算出** するといい、$u \triangleright v$ と記述する。算出されたバージョンは以下の対で表される。

$$u \triangleright v \Leftrightarrow v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \{u\} \rangle$$

ここで $\boldsymbol{E}_v$、$\boldsymbol{D}_v$、$\mathcal{T}$ は各々シミュレーションの対象事象、結果、完了時刻である[*1]。

2 つのバージョン u、v の **合同** $(u \oplus v)$ とは、2 つのバージョンに含まれるデータを合わせたものを表すバージョンである。合同バージョンは仮想的なバージョンであり、事象集合やデータ集合は空集合として扱われる。すなわち、u と v の合同バージョンは以下の対で表される。

$$u \oplus v = \langle \phi, \mathcal{T}, \phi, \{u, v\} \rangle$$

ただし、$\mathcal{T}$ は合同の操作を行った時刻である。

バージョン v がバージョン u を**直接根拠**とする $(u \succ v$ と記述) とは、v が u から合同あるいは算出されている場合をさす。上で定義されているように、v とその直接根拠 u には $u \succ v \Leftrightarrow u \in \boldsymbol{G}_v$ という関係が成り立つ。ただし、$v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$ である。

バージョン u がバージョン v の**根拠**である $(u \succsim v)$ とは、u と v の間に再帰的に直接根拠の関係のパスがあることをさす。すなわち、

$$u \succsim v \Leftrightarrow (u = v \text{ もしくは } u \succ u', u' \succsim v)$$

である。また、バージョン v の根拠の集合を v の**軌跡** $(\mathbf{Tr}(v) = \{u | u \succsim v\})$ と呼ぶ。

*1 ここでは、シミュレーションの結果はすぐさまデータベースに記録されると仮定している。

3.4 整合性

2 つのバージョン u, v が *1* **次的に無矛盾** $(u \tilde{\sim} v)$ であるとは、以下の条件を満たすときである。

$$\forall e \in \boldsymbol{E}_u \cap \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e$$

ただし、$u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$、$v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$ である。

2 つのバージョン u, v が **無矛盾** $(u \sim v)$ であるとは、以下の条件を満たすときである。

$$\forall u' \succsim u, \forall v' \succsim v : u' \tilde{\sim} v'$$

2 つのバージョンの合同 $u \oplus v$ をつくる場合、u と v は無矛盾である必要がある。

また、$v \sim v$ が成り立つ場合、バージョン v が **自己無矛盾**であるという。

3.5 世界線

ある事象に対して複数のデータがデータベースに登録されていることがあり得る。一方、事象の解析やシミュレーションでは、それらのデータを全て真として扱うとは限らず、目的に応じて取捨選択された一部のデータを真して解析などの入力とする。ここでは、一連のシミュレーションにおいて真として扱われたデータの集合を**世界線**と呼ぶ。

シミュレーションでは一般に異なる初期値を用いた解析を行うので、世界線も複数存在し得る。このような場合、異なる初期値を用いていることから、それらの世界線は相互に矛盾する。一方、一つの世界線の中ではそれに含まれる全てのデータは相互に無矛盾である必要がある。ここで無矛盾とは「各々のデータが相互に矛盾しない条件あるいはデータに基づいて求められたものである」という状態をさす。以下ではこれらの概念を形式的に定義する。

世界線 $\boldsymbol{w}$ はデータベース **DB** の任意の部分集合である。すなわち $\boldsymbol{w} \subseteq \mathbf{DB}$ である。世界線 $\boldsymbol{w}$ が同じ事象 e に対し複数のデータを含んでいる場合はモンテカルロ的に解釈され、その複数のデータは e の値の分布に準じていると見なす。一方、世界線がある事象についてデータを含んでいないときは、その事象については考慮しない ('don't-care') ものとして扱う。

世界線 $\boldsymbol{w}$ のうち、ある時刻 t に関する事象のデータのみを集めたものを時刻 t における世界線 $\boldsymbol{w}$ の**射影**と呼び、$\boldsymbol{w}/t$ と表す。すなわち、$\boldsymbol{w}/t = \{d_{\theta,t',s;\tau} \in \boldsymbol{w} | t' = t\}$ である。また世界線 $\boldsymbol{w}$ のうち、ある事物 θ に関するデータのみを集めたものを事物 θ における世界線 $\boldsymbol{w}$ の**射影** と呼び、$\boldsymbol{w}/\theta$ で表す。すなわち、$\boldsymbol{w}/\theta = \{d_{\theta',t,s;\tau} \in \boldsymbol{w} | \theta' = \theta\}$ である。同様に世界線 $\boldsymbol{w}$ のうち、ある事象 e に関するデータのみを集めたものを事象 e における世界線 $\boldsymbol{w}$ の**射影** と呼び、$\boldsymbol{w}/e$ で表す。すなわち、$\boldsymbol{w}/e = \{d_{e',s;\tau} \in \boldsymbol{w} | e' = e\}$ である。

以下の条件を満たした場合、またそのときに限り世界線 $\boldsymbol{w}$ がバージョン v を**支持**すると呼び、$\boldsymbol{w} \vdash v$ と記述する。

$$\forall u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u \rangle \succsim v, \forall e \in \boldsymbol{E}_u : \boldsymbol{w}/e = \boldsymbol{D}_u/e$$

また、以下の条件を満たした場合、またそのときに限り、世界線 $\boldsymbol{w}$ が **無矛盾**であると呼ぶ。

$$\forall d_{\theta,t,s;\tau} \in \boldsymbol{w} : s \text{ is a sensor id or } \boldsymbol{w} \vdash s$$

言い換えれば、世界線の中でシミュレーションで求められたデータは、それが属するバージョンの根拠が全て世界線に含まれている場合に限り、世界線は無矛盾となる。

一般にシミュレーションは世界線を伸ばしていく作業と言える。ここで重要なのは、世界線が無矛盾性を維持するようにシミュレーションを行わなくてはならないことである。もし世界線が無矛盾に保てないとすると、そのシミュレーションは異なる相互に相容れない根拠に基づいて計算を行っていることになり、全体としては意味のない計算を行うことになる。

3.6 無矛盾性に関する定理

ここでは、世界線およびバージョンの無矛盾性の維持に関していくつかの定理を与える。

補題 1. *バージョン v が自己無矛盾であり、かつ、世界線 $\boldsymbol{w}$ が v の軌跡に属する全てのデータから構成されている場合、v の任意の根拠 u は世界線 $\boldsymbol{w}$ に支持される。*

証明 *(補題 1).* バージョン v の根拠の 1 つ u が世界線 $\boldsymbol{w}$ に支持されていないとする。すなわち、

$$\exists u = \{\boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u\} \not\succ^* v, \exists e \in \boldsymbol{E} : \boldsymbol{w}/e \neq \boldsymbol{D}_u/e.$$

とする。定義より、$\boldsymbol{w}$ は $\boldsymbol{D}_u$ の上位集合である。よって、$\exists d \in \boldsymbol{w}/e, d \notin \boldsymbol{D}_u/e$ が満たされる限り $\boldsymbol{w}/e \neq \boldsymbol{D}_u/e$ が成り立つ。これは、

$$\exists u'\{\boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'}\} \in \mathbf{Tr}(v) \quad : \quad u' \neq u, \\ d \in \boldsymbol{D}_{u'}$$

である。しかしこれはバージョン v の自己無矛盾性の定義に反する。

よって、根拠 u は $\boldsymbol{w}$ に支持される。 □

この補題より、次の定理が導き出される。

定理 1. *バージョン v が自己無矛盾である場合、それを支持する無矛盾な世界線 $\boldsymbol{w}$ が存在する。*

証明 *(定理 1).* 次のような世界線 $\boldsymbol{w}$ を考える。

$$\boldsymbol{w} = \{d | u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle \not\succ^* v, d \in \boldsymbol{D}_u\}$$

補題 1 より世界線 $\boldsymbol{w}$ はバージョン v の任意の根拠を支持する。一方、任意のシミュレーションされたデータ $d_{\theta,t,s;\tau} \in \boldsymbol{w}$ に対して、[*2]そのデータが属するバージョン s は v の軌跡に含まれる。よって、s は世界線 $\boldsymbol{w}$ に支持される。

よって、無矛盾であり、かつ、v を支持する世界線 w は存在する。 □

この定理により、シミュレーションが意味あるものとする、すなわち、シミュレーションの結果が無矛盾な世界線の一部となるためには、そのシミュレーションが出力するデータのバージョンの自己無矛盾性にのみ配慮すれば良いことになる。

この定理を元に、以下ではバージョンの生成に関する定理を導き出す。

定理 2. *発生させられたバージョンは自己無矛盾であり、それを支持する無矛盾な世界線が存在する。*

証明 *(定理 2).* 定義により、発生させられたバージョンに含まれるデータは全てセンサーから直接きたものであり、他のバージョンとの依存関係はない。よって、自己無矛盾である。

よって、定理 1 より、発生バージョンを支持する無矛盾な世界線が存在する。 □

*2 センサーから得られたデータは根拠を持たないので、無矛盾性には抵触しないため、考慮しない。

定理 3. *自己無矛盾なバージョン u からバージョン v を抽出する場合、以下の条件を満たす場合に限り、v は自己無矛盾であり、それを支持する無矛盾な世界線が存在する。*

$$\forall v : u \supset v,$$
$$v \textit{ is self-consistent} \leftrightarrow \forall e \in \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e$$

ただし、u、v は各々$\langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$ および $\langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$ である。

証明 *(定理 3).* もし上記の条件が満たされている場合、u and v は無矛盾である。また、u は自己無矛盾であることから、v もまた自己無矛盾である。

逆に、条件 $\forall e \in \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e$ が満たされない場合、v に含まれるある事象 e が存在して、かつ、$\boldsymbol{D}_u/e \neq \boldsymbol{D}_v/e$ を満たすものが存在する。よって、u と v は矛盾することになり、v は自己無矛盾でなくなる

よって、v の自己無矛盾性と上記の条件は等価であり、定理 1 より、抽出バージョンを支持する無矛盾な世界線が存在する。 □

定理 4. *バージョン u が自己無矛盾である場合、それから算出されたバージョン v は自己無矛盾であり、v を支持する無矛盾な世界線が存在する。*

証明 *(定理 4).* 定義により、$\mathbf{Tr}(v)$ is $\{v\} + \mathbf{Tr}(u)$ が成り立つ。また、$\boldsymbol{E}_u \cap \boldsymbol{E}_v = \phi$ であることから、u の任意の根拠は v と無矛盾である。よって、バージョン v は自己無矛盾である。

よって、定理 1 から、抽出バージョンを支持する無矛盾な世界線が存在する。 □

定理 5. *2 つのバージョン v、u が自己無矛盾でありかつ相互に無矛盾であるとする。この場合、この 2 つのバージョンの合同バージョンは自己無矛盾であり、それを支持する無矛盾な世界線が存在する。*

証明 *(定理 5).* 定義より、

$$\mathbf{Tr}(u \oplus v) = \{u \oplus v\} + \mathbf{Tr}(u) + \mathbf{Tr}(v)$$

である。x と y を $\mathbf{Tr}(u \oplus v)$ に含まれる 2 つのバージョンとする。もし x と y が共に $\mathbf{Tr}(u)$ に含まれている場合、u が自己無矛盾であるため、x と y は無矛盾である。x と y が共に $\mathbf{Tr}(v)$ に含まれている場合も同様である。

$x \in \mathbf{Tr}(u)$、$y \in \mathbf{Tr}(v)$ かつ、$x \notin \mathbf{Tr}(v)$、$y \notin \mathbf{Tr}(u)$ である場合、u と v は相互に無矛盾であるので、x と y は相互に無矛盾である。

x と y のいずれかが $u \oplus v$ と一致する場合、$u \oplus v$ が事象を含まないことから、x と y は 1 次的似無矛盾である。

よって、合同バージョン $u \oplus v$ は自己無矛盾であり、定理 1 よりそれを支持する無矛盾な世界線が存在する。 □

4. バージョン管理のコスト

前節で議論した無矛盾性を保ちつつ統合シミュレーションを行うためには、データベースに以下のような機能を追加する必要がある。

- バージョンに関する情報 $\langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \boldsymbol{G} \rangle$ を記録する機能。特に、$\boldsymbol{D}$ は多数に上ることを前提に扱われることが必要。

- 既存のバージョンの無矛盾性を保ちつつ、新しいデータを登録する機能。特に、シミュレーション開始後にそのシミュレーションに関係するデータが挿入された場合、関係するバージョンの無矛盾性を管理する方法が必要。
- 2 つのバージョンの無矛盾性をチェックする機能。前節の定理にあるように、バージョンの合同を行う場合、その無矛盾性を高速にチェックする機能が必要。

以下ではこれらの機能を実装する方法として、$\boldsymbol{D}$ の記録方法に着目し、**正記法** と **負記法** の 2 つの方法とその計算コストについて考察する。

4.1 正記法

あるバージョンに関する $\boldsymbol{D}$ の最も単純な記録方法は、それに含まれるデータを列挙する方法である。これを **正記法** と呼ぶ。

この正記法ではデータの数に比例した記憶容量 ($O(|\boldsymbol{D}|)$) が必要となる。これは、災害救助のような大規模なシミュレーションではかなりのコストになる。

また、新しいデータを $\mathbf{DB}$ にいれる場合、その新しいデータはこれまで生成されたバージョンには全く含まれないため、正記法では特に余計なコストは生じない。

前節の定理で述べているように、シミュレーションにおいて成長する世界線を無矛盾に保つためには、バージョンの合同を行う際の無矛盾性の確保が一番問題となる。2 つのバージョン u と v の無矛盾性をチェックするためには以下の操作が必要である。

$$\forall u' \in \mathbf{Tr}(u), \forall v' \in \mathbf{Tr}(v), \forall e \in (\boldsymbol{E}_{u'} \cap \boldsymbol{E}_{v'}) : \\ \text{check } \boldsymbol{D}_{u'}/e = \boldsymbol{D}_{v'}/e$$

この操作は $O(\sum |\boldsymbol{D}_{u'}| + \sum |\boldsymbol{D}_{v'}|)$ の計算量を必要とする。

4.2 負記法

$\boldsymbol{D}$ のもう一つの記録方法として、そのバージョンが関係している事象 $\boldsymbol{E}$ のなかで使われなかったデータを記録する方法が考えられる。これを **負記法** と呼ぶ。

一般に、使われなかったデータは以下の場合に生じる。

- 1 つの事象に対し異なる値が報告されており、その一方を真、他方と偽として扱ってシミュレーションを行う場合。
- あるシミュレーションが時刻 $\mathcal{T}$ に開始され、その後、そのシミュレーションに関係する事象のデータが報告された場合。

負記法では以下のデータ集合を記録することになる。

$$\bar{\boldsymbol{D}}_v = \sum_{e \in \boldsymbol{E}_v} (\mathbf{DB}_{(0,\mathcal{T}_v)}/e - \boldsymbol{D}_v)$$

ここで注意しておかなければならないのは、上の式で $\mathbf{DB}$ の変わりに $\mathbf{DB}_{(0,\mathcal{T}_v)}$ を用いていることである。つまり $\bar{\boldsymbol{D}}_v$ は時刻 $\mathcal{T}_v$ において使われなかったデータのみを記録する。これ以降に入ってきたデータについては、そのデータのタイムスタンプ $\mathcal{T}$ とこのバージョンのタイムスタンプ $\mathcal{T}_v$ を比較することで $\boldsymbol{D}_v$ に含まれるかどうかが判定できる。

負記法による記録容量は $O(|\bar{\boldsymbol{D}}|)$ である。一般に関連する事象で使われないデータは、使われるデータより少ないため、正記法よりは効率が良いと考えられる。

新しく登録されるデータは既存のバージョンでは使われなかったデータであるので、負記法では本来ならばいちいち $\bar{\boldsymbol{D}}_v$ に追加する必要がある。しかし上で述べているように、データが記録された時刻とバージョンのタイムスタンプの比較を行うことで $\boldsymbol{D}_v$ に含まれているかどうかを容易に判定できるため、実際には余計なコストは生じない。

2 つのバージョン u と v の無矛盾性をチェックするためには以下の操作が必要である。

$$\forall u' \in \mathbf{Tr}(u), \forall v' \in \mathbf{Tr}(v), \forall e \in (\boldsymbol{E}_{u'} \cap \boldsymbol{E}_{v'}) : \\ \begin{cases} \text{check } \bar{\boldsymbol{D}}_{v'}/e - \bar{\boldsymbol{D}}_{u'}/e = \mathbf{DB}_{(\mathcal{T}'_u,\mathcal{T}'_v)}/e & \text{if } \mathcal{T}'_u < \mathcal{T}'_v \\ \text{check } \bar{\boldsymbol{D}}_{u'}/e - \bar{\boldsymbol{D}}_{v'}/e = \mathbf{DB}_{(\mathcal{T}'_v,\mathcal{T}'_u)}/e & \text{if } \mathcal{T}'_v < \mathcal{T}'_u \end{cases}$$

ただし、$\mathbf{DB}_{(\mathcal{T}_x,\mathcal{T}_y)}$ は時刻 $\mathcal{T}_x$ から 時刻 $\mathcal{T}_y$ の間に記録されたデータの集合である。この操作の計算コストは、$O(\sum |\mathbf{DB}_{(\mathcal{T}_{v'},\mathcal{T}_{u'})}/\boldsymbol{E}| + \sum |\bar{\boldsymbol{D}}_{u'}| + \sum |\bar{\boldsymbol{D}}_{v'}|)$ である。

5. 補足的な議論

正記法と負記法のどちらが効率がよいかは一般に決めることはできない。しかし大規模シミュレーションでは多くの場合、データの数は非常に多くなるため、前節の考察では負記法の方が記憶容量の点で有利と思われる。

また、4. 節では射影の操作 (例えば $\mathbf{DB}_{(\mathcal{T}'_u,\mathcal{T}'_v)}/e$) を無視している。これは、一般的な RDB を用いた場合、これらの操作は $\log N$ 程度の高速な手法が用意されていることを前提としているためである。

また、事象集合 $\boldsymbol{E}$ の管理コストも上記の議論では無視している。これは、シミュレーションに関係する事象というのは、一般に機械的に指定されることを前提としている。例えば災害シミュレーションでは、入力としては指定された領域の特定の種類の地物 (「建物」や「道路」、「震度分布」など) という形で与えられる。これらをリストアップすることも可能だが、汎用的なデータベースを用いていれば、必要に応じて検索により求めていてもそれほどコストは生じないと考えられる。

この他、以下のような問題が形式化の上で残っている。

- バージョンの抽出に関する計算コスト。
- 統計的処理の合同や、モンテカルロシミュレーションの結果の解析に対するバージョンの考え方。

参考文献

[Tad03a] Satoshi Tadokoro. Japan government special project: development of advanced robots and information systems for disaster response (daidaitoku) — for earthquake disaster mitigation in urban areas —. In *Proc. of IEEE Workshop on Safety Security and Rescue Robotics (SSRR03)*, 2003.

[Tad03b] Satoshi Tadokoro. An overview of japan national special project daidaitoku (ddt) for earthquake disaster mitigation in urban areas. In *Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003) Workshop on Advanced Robots and Information Systems for Disaster Response*, 2003.