Vol.2010-GN-75 No.2
2010/3/18

制約充足モデルに基づく Web コンテンツ管理システム

村崎大輔[†1]　橋田浩一[†1]

制約に基づくワークフローなどの記述は，要求仕様と実装とのギャップを埋める有効な手段である．制約充足モデルは独立したロジック層として実装されるため，制約充足とプレゼンテーション層のユーザインタフェースとを同期させる手法が重要となる．Web アプリケーションをサポートするコンテンツ管理システム CBCMS において，RDF ベースのデータを対象とした制約伝搬とリアルタイムに同期するプレゼンテーション層を簡潔に実装するための手法を提案する．

A Web Content Management System Based on Constraint-Satisfaction Models

Daisuke Murasaki[†1] and Koiti Hasida[†1]

Constraint-based models of workflows and other social interactions are effective for filling the gap between specifications and implementations. Since a constraint satisfaction model is implemented as an independent logic layer, it is important to synchronize results of constraint processing and user interfaces in the presentation layer. We propose how to concisely describe the presentation layer to allow real-time reflections of the constraint propagation across RDF data to the presentation layer. This technique shall be incorporated in CBCMS, a content management system to support Web-based applications.

1. はじめに

ワークフローなどを含む社会的相互作用を情報システムとして実装する際において，要求仕様と実装との間に存在するセマンティックギャップが問題となっている．この問題に対して，業務処理の要求仕様を制約に基づいて簡潔に記述してビジネスロジックを制約充足によって処理する手法が提案されている．例えば，制約に基づくビジネスロジックの処理系の一つに CBTO (Compositional Business-Task Organization)[1),2)] が提案されているほか，制約充足モデルに基づいてワークフローを記述する手法として 3) がある．

CBTO では業務処理が完了した状態を制約で表現することにより，業務処理が正しくなされているかどうかを制約によって検証しつつ，制約が充足されるまで処理が進行する．例えば差し戻し等を手続き的な手法で記述すると実装が複雑になる傾向があるが，CBTO では業務処理の状態を制約が満たされない状態に遷移させることで差し戻し処理を実現できる．

業務処理の概念をオントロジーによって構造化することで，業務処理は基本的にすべて制約充足問題として記述することができる．一般的な制約処理エンジンに帰着されない制約処理もあるが，それらを個別に実装することでビジネスロジック全体が実装される．その結果として制約によって記述された要求仕様とソフトウェアの実装をスムーズに対応づけることができ，セマンティックギャップを解消し，既存のシステムに比べてビジネスロジックの実装や改修を容易に行えることが期待される．

CBTO の処理系は独立したロジック層として実装されるため，ビジネスロジックの実行によって生じた制約充足の結果とプレゼンテーション層のユーザインタフェースを同期する方法 (特に前者を後者に反映させる手法) が重要となる．そこで，本稿では Web アプリケーションのバックエンドで動作するコンテンツ管理システムを対象として，制約伝搬などによって生じるデータモデルの変更をリアルタイムに反映するプレゼンテーション層を簡潔に記述するための手法を提案する．

2. セマンティックオーサリング

業務処理を CBTO によってシステム化する際には，対象となる業務処理の概念や情報をオントロジーによって構造化し，構造を明示したコンテンツとして作成することが重要である．このようにオントロジーに基づいて構造化されたコンテンツを作成することをセマンティックオーサリング (Semantic Authoring)[4)] と呼ぶ．

セマンティックオーサリングを実現する手法はいくつかあるが，ここでは産総研が開発しているセマンティックエディタや SACM (Semantic Authoring Content Manager) などのアプリケーションを用いたコンテンツ作成を対象とする．

†1 産業技術総合研究所 社会知能技術研究ラボ
Social Intelligence Technology Research Laboratory, National Institute of Advanced Industrial Science and Technology (AIST)

Vol.2010-GN-75 No.2
2010/3/18

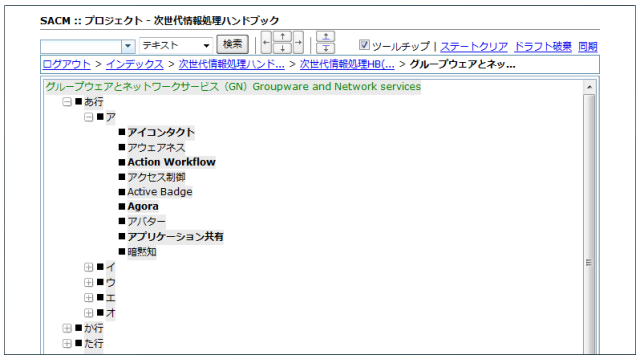

図 1 SACM
Fig. 1 SACM.

2.1 セマンティックエディタのデータモデル

セマンティックエディタは RDF (Resource Description Framework) のグラフ構造などを共同作成するための Java アプリケーションである．セマンティックエディタのデータは RDF のトリプルを基にしながらも，CDL (Concept Description Language) のハイパーノード (W3C 用語では名前付グラフ) や，ノードに付随する補助的な情報も扱うことができる．これらの情報は本質的に RDF グラフと同じであるため，以降は包括的に「RDF コンテンツ」と呼ぶ．

セマンティックエディタでは，グラフィカルな編集に必要なユーザインタフェースに求められる機能にも重点が置かれている．ハイパーノードは要素コンテンツのリストとレイアウト情報を持つ．要素コンテンツが構成するグラフはグラフ表示やツリー表示によってセマンティックエディタのウィンドウに表示される．

SACM はセマンティックエディタの Web アプリケーション版で，図 1 のようなツリー表示による編集インタフェースを持つ．SACM は情報処理学会創立 50 周年記念事業の一環として制作されている「次世代情報処理ハンドブック」の編纂にも利用されている．

セマンティックエディタで扱う RDF コンテンツにプレゼンテーション層からアクセスする場合，RDF のグラフ構造だけでなく，ハイパーノードの構造やレイアウト情報などにアクセスするための手法が必要である．

2.2 制約に基づくモデル化の例

制約に基づくモデル化の例として，健康診断システムにおける身体測定のデータを取り上げる．身体測定のオントロジーは，例えば図 2 のように記述することができる．

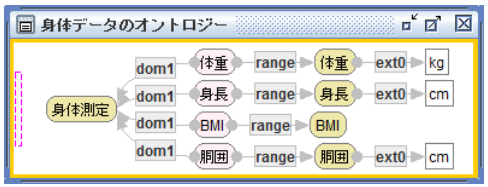

図 2 身体測定のオントロジー
Fig. 2 Body measurement ontology.

図 3 身体測定のインスタンス
Fig. 3 Instance of body measurement.

角丸長方形の「身体測定 (`health:BodyMeasurement`)」ノードは RDF/OWL のクラスを表しており，`health:BodyMeasurement` はセマンティックエディタ内部で扱うためのシンボル名である．「身体測定」から `dom1` プロパティを介して結ばれたノードは RDF/OWL にはないプロパティであり，「体重 (`health:weight`)」，「身長 (`health:height`)」，「BMI (`health:bmi`)」，「胴囲 (`health:waist`)」の各プロパティのインスタンスであるリンクが，「身体測定」クラスのインスタンスであるノードを始点 (subject) としてちょうど 1 つだけ現われることを意味する．これらのノードから `range` プロパティを介して結ばれたノードはクラスであり，各プロパティの終点 (object) となるノードが，それぞれ「体重 (`health:Weight`)」，「身長 (`health:Height`)」，「BMI (`health:BMI`)」，「胴囲 (`health:Waist`)」クラスのインスタンスであることを表している．このオントロジーに基づいて，身体測定のインスタンスは図 3 のように表現できる．ここで，胴囲の情報は BMI の制約とは関係ないため省略した．

BMI は 体重 $\div$ 身長^2 として求められることから，身長と体重を入力変数，BMI を出力変数とした単方向の制約を定義することができる．身長体重と BMI の関係が制約として定義されると，制約充足エージェントは制約伝搬などの手法を用いながら，変数の変化に応じて制約を満たす値を求めていく．例えば BMI を出力変数とした単方向の制約処理では，身長や体重の値が変化した場合に BMI の値を再計算し，制約が満たされるようにする．

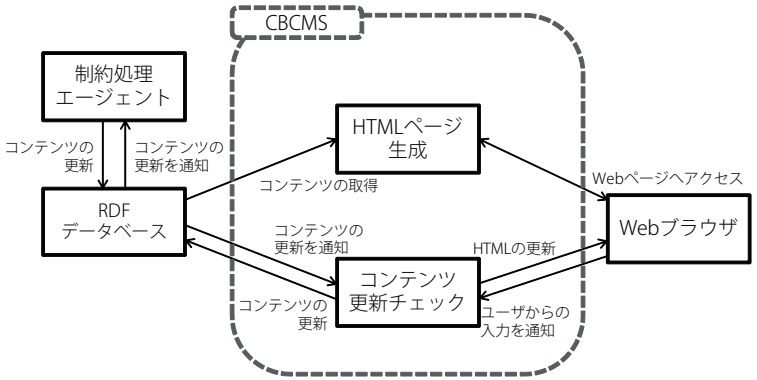

図 4 CBCMS の概要
Fig. 4 Overview of CBCMS.

3. CBCMS

本稿では，上述した RDF コンテンツを扱いつつ，制約伝搬の結果をリアルタイムに反映することが可能な Web コンテンツ管理システム (CMS) として，CBCMS (Constraint-Based Content Management System) を提案する．

3.1 CBCMS の構成

CBCMS の機能は図 4 にあるように大きく 2 つに分類できる．一つは RDF コンテンツを用いながら HTML ページを生成する機能であり，もう一つは制約伝搬の結果に追従して生成された HTML ページの内容を書き換える更新チェックサーバである．

CBCMS が生成する HTML ページは RDF コンテンツと HTML の断片が一対一で対応するようにし，その対応づけられた部分を最新の状態に保つための処理は CBCMS が提供する．これは Microsoft .NET Framework の Windows Presentation Foundation (WPF) が提供しているデータバインディング (Binding) と類似したアプローチである．

ノードの情報に何らかの操作を適用したり，条件を満たすノードの総数を返したりするような集約演算は，全て制約充足エージェントなどを介するようにしてロジック層で行うようにする．これにより，更新すべき内容を容易に対応付けることができる．

CBCMS を実装するにあたっては，既存の CMS が持っている資源を最大限に活用できるような形が望ましい．そこで CBCMS は Radiant CMS[5] の拡張機能として実装する．

3.2 CBCMS における HTML ページ生成

CBCMS では RDF コンテンツから HTML ページを作成するために，以下の 2 つの手法を提供する．

- HTML ページの内容を RDF コンテンツとして記述し，CBCMS がコンテンツを解釈して HTML ドキュメントを出力する手法
- HTML ページの内容を HTML と親和性の高いテンプレート言語によって記述し，CBCMS がテンプレートを解釈しながら RDF コンテンツを埋め込み，HTML ドキュメントを出力する手法

これらの記述手法は相互補完的に用いることを想定している．すなわちロジック層で変化しうる HTML ドキュメントの断片を RDF コンテンツとして作成し，その内容を埋め込むページを CMS が提供しているテンプレート言語で記述することができる．

これにより，双方の利点を生かしながらプレゼンテーション層のユーザインタフェースを作成することができる．特に HTML ページのデザインに関する部分には CSS (Cascading Style Sheets) を用いるようにし，デザインにかかるコストの低減を図る．

3.3 HTML ページの RDF コンテンツによる記述手法

まず，HTML ページの内容を RDF コンテンツとして記述する手法について述べる．セマンティックエディタではノードの型をオントロジーで定義されたクラスで指定するだけでなく，ノードが持つデータの種類を MIME タイプで指定することができる．

最も単純なアプローチとして，MIME タイプを `text/html` としてノードのデータに HTML テキストを格納する手法が考えられる．しかしながら，この手法では複数のノードから構成される内容を HTML に含めることが困難であるため，RDF ベースの制約充足モデルを扱うには不十分であり，CBCMS のページ生成手法には適さない．

そこで CBCMS では，セマンティックエディタが持つレイアウト情報を活用し，HTML の構造を木構造に細分化して格納する．HTML を木構造にして表現した例を図 5 に示す．図 5 でラベルが `h1`, `p`, `br`, `input` となっている灰色のノードは，HTML のタグを表している．HTML タグの構造はセマンティックエディタのツリー表示の構造と対応している．

CBCMS は図 5 の内容を解釈し，図 6 に示す HTML を出力する．出力された HTML ではノードのラベルとなっているテキストがタグで囲まれている．これはノードの内容が変更されたとき，後述する更新チェックにおいて書き換えるべきテキストを特定するための処理である．

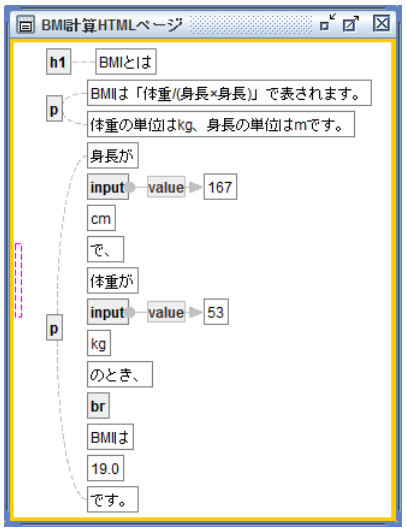

図 5 HTML ページコンテンツ
Fig. 5 HTML page content.

```
<h1>BMI とは</h1>
<p>
  <span id="...">BMI は「体重/(身長×身長)」で表されます。</span>
  <span id="...">体重の単位は kg、身長の単位は m です。</span>
</p>
<p>
  <span id="...">身長が</span>
  <input id="..." value="167">
  <span id="...">cm</span>
  <span id="...">で、</span>
  <span id="...">体重が</span>
  <input id="..." value="53">
  <span id="...">kg</span>
  <span id="...">のとき、</span>
  <br>
  <span id="...">BMI は</span>
  <span id="...">19.0</span>
  <span id="...">です。</span>
</p>
```

図 6 図 5 の HTML 出力
Fig. 6 HTML output of Fig. 5.

3.4 SA セレクタ

テンプレート言語を用いて RDF コンテンツを埋め込むためには，処理対象のノードを指定するための手法が必要である．以降，これをセレクタと呼称する．

セレクタは十分な表現力を備え簡潔な記述が可能な構文を持つ必要がある．最も単純なセレクタとしてノードが持つ ID を直接指定する手法があるが，それでは対応関係が固定されてしまうので，RDF コンテンツに対する再入的 (reentrant) なパターンマッチが不可能である．

低コストにセレクタを実現するために，既存のクエリ言語やパターンマッチ言語の構文を取り入れるアプローチが考えられる．例えば W3C 勧告となっている構文として，SPARQL や XPath，CSS セレクタなどが挙げられる．しかしながら，これらの構文を CBCMS に適用させようとした場合にはデータモデルとのギャップが問題となる．SPARQL は RDF を対象としているためにセマンティックエディタのハイパーノードやレイアウト情報を直接扱えず，また汎用性の高い問い合わせ言語であるためにコンパクトな記述が難しい構文となっている．XPath や CSS セレクタは XML や HTML の木構造を対象としているため，RDF のグラフ構造を扱う場合には記述が冗長になる．コンテンツを XML に展開してから XPath などで処理するアプローチも考えられるが，中間的な処理が介在することでパフォーマンス上のデメリットが生じる．

RDF コンテンツに適した簡潔な記述を可能にするため，CBCMS では対象ノードを選択する手法として独自の SA セレクタを提供する．SA セレクタの構文は，CSS2 のセレクタを参考にしつつ RDF 等のコンテンツに適したものとした．SA セレクタの構文の詳細については付録 A.1 を参照されたい．

SA セレクタはカレントノードを起点とし，そこからパターンマッチを繰り返すことで対象となるノードを選択していく．複数のパターンがスペース区切りで含まれている場合，前のパターンにマッチしたノードに対して次のパターンのマッチを行っていき，全てのパターンにマッチするノードが選択される．

SA セレクタのパターンマッチによって選択される複数のノードは順序を持つ．この順序はセマンティックエディタのツリー表示におけるノードの出現順序に対応する．

3.5 SA タグを用いたテンプレート記述

Radiant CMS では Radius と呼ばれるタグベースのテンプレート言語が提供されており，Radiant CMS の拡張機能では独自の Radius タグを定義することができる．そこで，CBCMS では RDF コンテンツを埋め込むための SA タグを提供する．

SA タグは sa タグを親としてネストしたタグとして定義する．後述する sa:find タグの場合，<r:sa:find></r:sa:find>と記述しても<r:sa><r:find></r:find></r:sa>と記述してもよい．ここで，タグ名の先頭に付いている r:は Radius タグを意味するプレフィックスである．

SA タグの詳細については付録 A.2 に示すが，ここではページ断片の生成を行うための sa:content タグと，SA セレクタで選択されたノードに対する繰り返し処理を記述するための sa:find タグの機能について述べる．

3.5.1 sa:content タグを用いたページ断片の生成

sa:content タグはカレントノードの内容や型情報を参照し，それに対応したテンプレートを用いて内容を展開する．テンプレートは Radiant CMS が提供している snippet として格納する．例えば図 3 で示した身体測定のノードをカレントノードとして sa:content タグが評価されると，health:BodyMeasurement という名称の snippet が持つ内容が展開される．

3.5.2 sa:find タグを用いた繰り返し処理の記述

sa:find タグはカレントノードから SA セレクタよって選択されたノードを取得し，各ノードをカレントノードとして sa:find:each タグで囲まれた部分を処理する．

sa:find:if_empty タグは sa:find タグで指定した SA セレクタに該当するノードが存在しなかった場合にタグで囲まれた部分を返し，存在した場合には空文字列を返す．sa:find:unless_empty タグは該当するノードが存在した場合にタグで囲まれた部分を返し，存在しない場合は空文字列を返す．

繰り返し処理の記述例として，身体測定の計測データを図 7 に示すようなリストとして記述し，制約名 cbcms:List, cbcms:item, cbcms:count を用いて計測データの総数に制約を持たせる場合について考える．このようなリストに制約ベースの手法で新しい計測データを追加するためには，総数の値を 1 つ増やすことで，その制約解消として新しい部分グラフを作成するようにする．

図 7 のコンテンツを Web ブラウザに図 8 のような表として表示するテンプレートの例を図 9 に示す．このテンプレートを CBCMS が処理すると図 10 に示すような HTML が出力される．紙面の都合上，図 10 では図 6 に含まれているタグの id 属性を省略した．

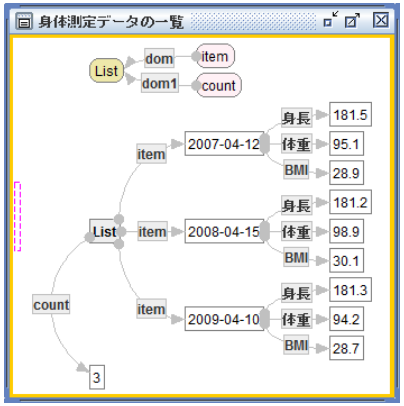

図 7 リストコンテンツ
Fig. 7 List content.

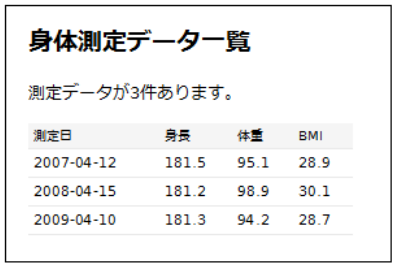

図 8 図 7 のコンテンツのレンダリング
Fig. 8 Rendering of Fig.7 content.

3.6 更新チェック

CBCMS では，SA セレクタでパターンマッチに用いたノードの変更を追跡することで，HTML ページとして出力された内容の更新が必要かどうかをチェックする．更新情報を Web ブラウザに通知する手段としては，AJAX デザインパターン[6] として述べられている HTTP Streaming (Comet) を利用する．更新チェックに必要な JavaScript コードの挿入は，テンプレート内に sa:updater タグを記述することによって実現する．また，HTML の input タグなどによってユーザがページ内の情報を変更させるような入力を実行した場合には，その内容を更新チェックのサーバに通知するようにする．

```
<h1>身体測定データ一覧</h1>
<r:sa:find_first p="cbcms:List cbcms:count >">
  <p>測定データが<r:sa:label />件あります。</p>
</r:sa:find_first>

<r:sa:find p="cbcms:List cbcms:item >">
  <r:unless_empty>
    <table>
    <thead>
    <tr><th>測定日</th><th>身長</th><th>体重</th><th>BMI</th></tr>
    </thead>
    <tbody>
  </r:unless_empty>

  <r:each tag="tr">
    <td><r:sa:label /></td>
    <td><r:sa:o:health:height /></td>
    <td><r:sa:o:health:weight /></td>
    <td><r:sa:o:health:bmi /></td>
  </r:each>

  <r:unless_empty>
    </tbody>
    </table>
  </r:unless_empty>
</r:sa:find>
```

図 9 表組みのテンプレート
Fig. 9 Table template.

```
<h1>身体測定データ一覧</h1>
<div id="...">
  <p>測定データが<span>3</span>件あります。</p>
</div>

<div id="...">
  <table>
  <thead>
  <tr><th>測定日</th><th>身長</th><th>体重</th><th>BMI</th></tr>
  </thead>
  <tbody>
  <tr id="...">
    <td><span>2007-04-12</span></td><td><span>181.5</span></td>
    <td><span>95.1</span></td><td><span>28.9</span></td>
  </tr>
  <tr id="...">
    <td><span>2008-04-15</span></td><td><span>181.2</span></td>
    <td><span>98.9</span></td><td><span>30.1</span></td>
  </tr>
  <tr id="...">
    <td><span>2009-04-10</span></td><td><span>181.3</span></td>
    <td><span>94.2</span></td><td><span>28.7</span></td>
  </tr>
  </tbody>
  </table>
</div>
```

図 10 図 9 の HTML 出力 (整形したもの)
Fig. 10 HTML output of Fig.9 (formatted).

制約の処理によって生じるページ更新は，HTML のテキストのみが変化する場合と HTML の要素が増減する場合に分けることができる．ここで更新を追跡すべきノードの集合は，カレントノードの ID とテンプレートの対をキーとして一意に定まる．すなわちテンプレートの内容が不変であれば，内容を最初にレンダリングした時点で追跡すべきノードの集合を求めることができる．

例えば有限領域制約の処理では，図 5 に示したようなノードの値が変化することになる．その場合，図 6 に出力された HTML のを手がかりに，要素内のテキストを変更するための情報を通知するようにすればよい．

HTML の要素が増減する場合としては RDF のグラフ構造の変化がある．例えば図 7 のようなコンテンツで RDF コンテンツのグラフ構造が変化すると，SA セレクタで選択されるノードが変化する可能性がある．すなわち図 10 のような HTML 出力では，cbcms:item によってリンクされるノードを伴う部分グラフの増減に対応して，対応する行を追加したり削除したりする必要がある．

ここでグラフ構造の変化が影響する範囲は SA セレクタを用いている部分に限られているため，ノードの増減があった場合には sa:find タグや sa:find:each タグに対応する部分のみを更新するようにすればよい．

4. ま と め

本稿では，制約充足モデルに基づく Web コンテンツ管理システムとして，ロジック層である RDF コンテンツからプレゼンテーション層である HTML ページを生成する CBCMS を提案した．

今後の課題として，情報処理ハンドブックのコンテンツなどを対象にして，項目閲覧用のシステムや項目執筆のワークフローをサポートするプロトタイプシステムを作成していくことが挙げられる．プロトタイプの実装を通して，制約充足モデルに基づく Web コンテンツ管理システムに必要とされる機能について，本稿で検討しなかった部分を明らかにしていくことができると考えられる．特に更新チェック時に行われるノード変更の追跡や HTML の更新手法に関しては，大規模なサイトに対する大量のアクセスを踏まえた最適なプロトコルと実装について，詳細を検討していく必要がある．

参 考 文 献

1) 橋田浩一：オントロジーと制約に基づくセマンティックプラットフォーム，人工知能学会誌， Vol.21, No.6, pp.712–717 (2006).
2) Hasida, K., Izumi, N. and Mori, A.: CBTO: Compositional Business-Task Organization, *W3C Workshop on Declarative Models of Distributed Web Applications* (2007).
3) 石井 恵，金田重郎：制約充足エージェントに基づくワークフローの実現，電子情報通信学会論文誌 D-I， Vol.J81-D-I, No.5, pp.496–504 (1998).
4) Handschuh, S. and Staab, S.: Authoring and Annotation of Web Pages in CREAM, *Proceedings of the 11th international conference on World Wide Web*, pp.462–473 (2002).
5) 前田 剛：入門 Radiant CMS，秀和システム (2009).
6) Mahemoff, M.: *Ajax Design Patterns*, O'Reilly Media (2006).

付 録

A.1 SA セレクタの構文

SA セレクタのパターン構文を以下に示す．ここで␣は空白文字を示す．

- `symbol`
  カレントノードの子ノードないし接続されているリンクのうち，型のシンボル名が `symbol` であるノードやリンクにマッチする．ここで `symbol1␣symbol2` のように複数のシンボルをスペース区切りで記述すると，`symbol1` を評価してマッチしたノードに対して，`symbol2` を評価してマッチしたノードを選択する．
- `symbol;first`
  カレントノードの子ノードないし接続されているリンクのうち，型のシンボル名が `symbol` である最初のノードやリンクにマッチする．
  CSS の構文では疑似クラスを指定するためにコロン (:) が用いられているが，SA ではシンボル名に名前空間の区切り文字としてコロンを用いることが慣例となっているため，SA セレクタではセミコロン (;) を用いる．
- *
  カレントノードの全ての子ノードにマッチする．
- p␣>
  カレントノードを始点とし，リンクの型が p であるリンクの終点であるノードにマッチする．セマンティックエディタではリンクの終点が ▹ で表されるため，>を用いる．
- p␣<
  カレントノードを終点とし，リンクの型が p であるリンクの始点であるノードにマッチする．p が対称的なプロパティであれば，p␣<は p␣>と同義である．
- `[literal="text"]` または `[literal='text']`
  カレントノードが持つリテラルのタイプが `literal` で，かつ内容が `text` と等しいノードにマッチする．
- `[literal/="regexp"]` または `[literal/='regexp']`
  カレントノードが持つリテラルのタイプが `literal` で，かつ内容が正規表現 `regexp` とマッチするノードにマッチする．
- `["text"]` または `['text']`
  `[rdfs:label="text"]` にマッチするノードか，カレントノードが持つデータの MIME タイプがテキストかつ内容が `text` と等しいノードにマッチする．この記法は，セマンティックエディタ上で表示されるノードのテキストに対するマッチと対応している．
- `[/regexp/]`
  `[rdfs:label/="regexp"]` にマッチするノードか，カレントノードが持つデータの MIME タイプがテキストかつ内容が正規表現 `regexp` とマッチするノードにマッチする．この記法は `["text"]` と同様に，セマンティックエディタ上で表示されるノードのテキストに対するマッチと対応している．
- `::`
  カレントノードがハイパーノードである場合，その内部 (子) をカレントノードとする．

A.2 SA タグ

テンプレート記述に用いられる SA タグを以下に示す.

- <r:sa id="node-id">
  カレントノードを node-id として，タグで囲まれた部分を処理する．id が指定されなかった場合は，ページの URL からカレントノードを特定する.
- <r:sa:data>
  カレントノードのデータを返す．データの MIME タイプがテキストであればテキスト内容，画像といった他の MIME タイプであれば<img>などの適切な HTML タグを返す.
- <r:sa:literal type="typename">
  カレントノードのリテラルで，タイプが typename であるものの内容を返す.
- <r:sa:label>
  カレントノードのラベルを返す．ラベルはタイプが rdfs:label のリテラルを含んでいる場合はその内容，含んでいない場合は sa:data タグで返される内容である.
- <r:sa:type>
  カレントノードの型に対応するラベルを返す.
- <r:sa:link_to>
  カレントノードへのリンクを返す．例えばページへのリンクを次のように記述すると，
  <r:sa:link_to>BMI の計算ページ</r:sa:link_to>
  カレントノードへのリンクを求めて次のような出力を返す.
  BMI の計算ページ
- <r:sa:find p="selector">
  カレントノードを起点として SA セレクタ selector にマッチするノードを選択する.
- <r:sa:find:each tag="tagname">
  sa:find タグで選択されたノードをカレントノードとして，タグで囲まれた部分を処理する．処理された内容は<tagname>タグで囲まれる.
- <r:sa:find:if_empty>
  sa:find タグで選択されたノードが存在しなかった場合にタグで囲まれた部分を返し，存在した場合には空文字列を返す.
- <r:sa:find:unless_empty>
  sa:find タグで選択されたノードが存在した場合にタグで囲まれた部分を返し，存在しない場合は空文字列を返す.
- <r:sa:find_first p="selector">
  カレントノードを起点として SA セレクタ selector にマッチする最初のノードを選択する.
- <r:sa:o:symbolname>, <r:sa:s:symbolname>
  カレントノードを始点 (sa:o:symbolname) ないし終点 (sa:s:symbolname) としたリンク symbolname の先にある最初のノードのラベルを返す．これは sa:find_first タグに対する簡便表現である．例えば図 3 に示した「身体測定」ノードをカレントノードとした場合，<r:sa:o:health:weight />と記述すると「身体測定」ノードを始点とした「体重」リンクの終点ノードのラベルである「53」に展開される.
- <r:sa:updater>
  更新チェックを行うための JavaScript コードを返す.