National Institute of Information and Communications Technology

階層的サービス連携のための キャッシュ機構

独立行政法人情報通信研究機構
田仲正弘, 村上陽平

サービスコンピューティング研究専門委員会　第2回研究会

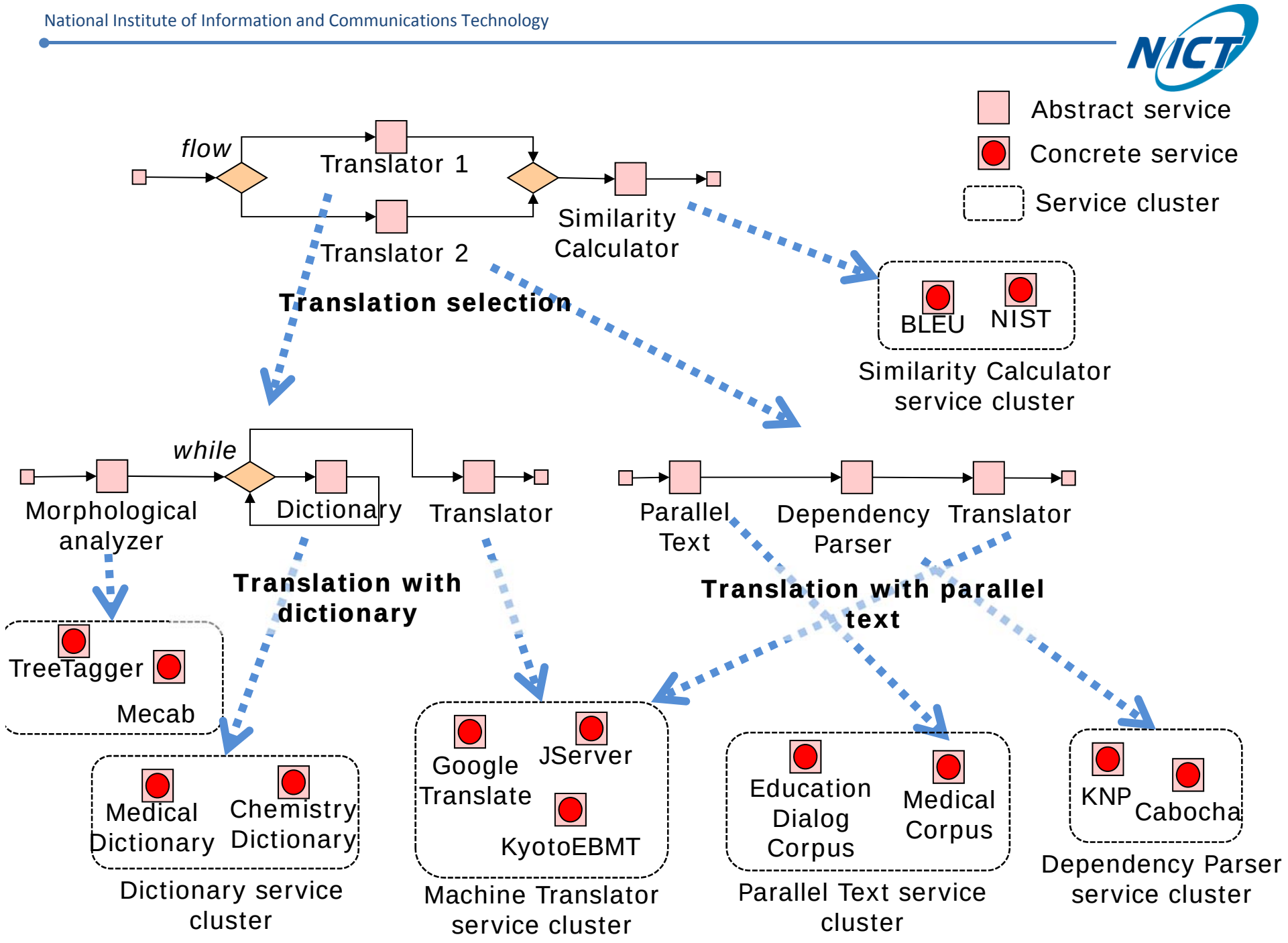
背景

- オープンなサービス環境：様々なサービス提供者／利用者が参加し，独自にサービスを提供／利用

- 例: 言語グリッド[1]
  - 18ヶ国から140以上の組織が参加
  - 170以上のサービスが登録 (機械翻訳, 辞書, etc.)
  - サービスを標準化

- 非機能的性質の設定のため，実行時に*サービスのバインディング*を決定

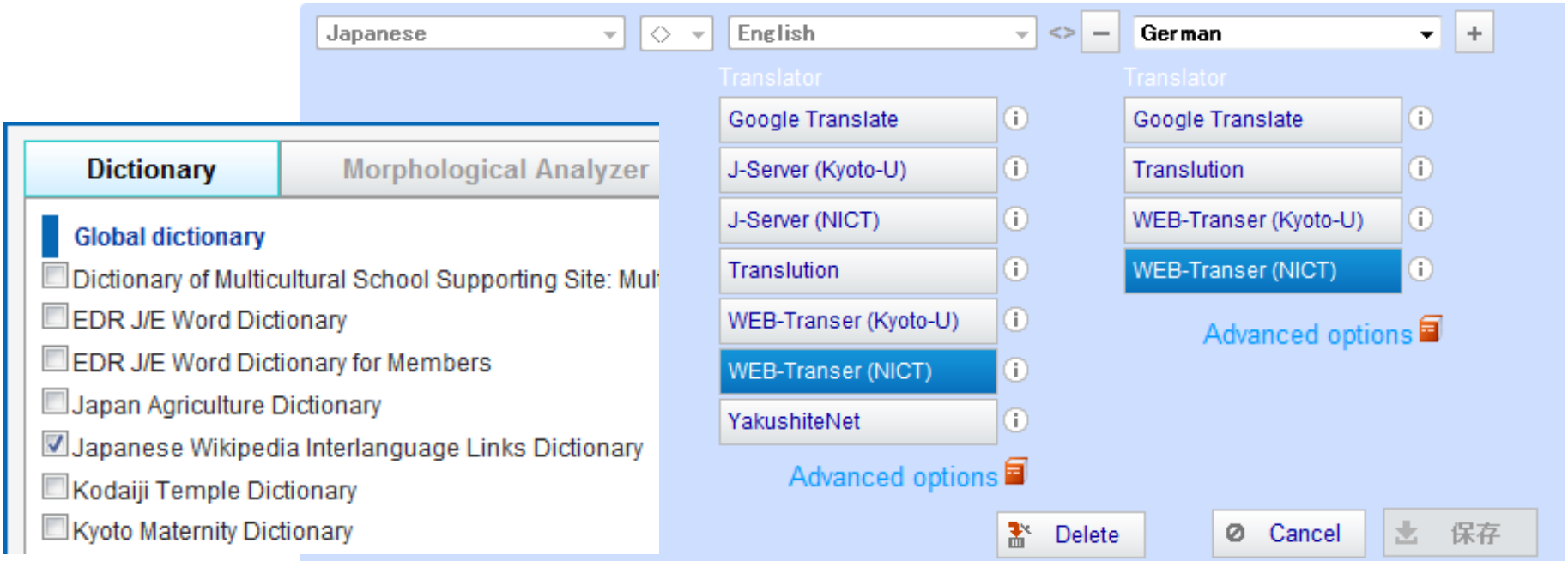
1 http://langrid.org


Abstract service
Concrete service
Service cluster
flow
Translator 1
Translator 2
Similarity Calculator
Translation selection
BLEU
NIST
Similarity Calculator service cluster
while
Morphological analyzer
Dictionary
Translator
Parallel Text
Dependency Parser
Translator
Translation with dictionary
Translation with parallel text
TreeTagger
Mecab
Medical Dictionary
Chemistry Dictionary
Dictionary service cluster
Google Translate
JServer
KyotoEBMT
Machine Translator service cluster
Education Dialog Corpus
Medical Corpus
Parallel Text service cluster
KNP
Cabocha
Dependency Parser service cluster

問題点1
階層的バインディング構成のコスト

- 各ユーザは興味のあるサービスのみ指定
- ポリシーに基づく制約は常に変化

→ システムが実行時に自動的にバインディングを構成する必要



辞書の選択

機械翻訳の選択

言語グリッド**Toolbox** (http://langrid.org/tools/toolbox)

問題点2
実行時適応の扱い

- 動的な環境の変化等に対応するため，様々な実行時適応の技術が提案
  - AO4BPEL[Charfi 07], Service Supervision[Tanaka 09], …
  - 複合サービス実行中に，新たなサービスの追加や一部サービスのスキップを可能にする

- 実行時適応が行われると，バインディングも再構築が必要
  → 実行時に実施すると性能が低下

アプローチ
ATMSの適用

- ATMS (Assumption-based Truth Maintenance System) [Kleer 88]
  - 仮説推論のためのフレームワーク
  - 複数のコンテキストの表現
  - 矛盾の効率的な管理

階層的サービス連携と仮説推論の対応

- 複数のサービスから一つのサービスを構成⇔ルールに基づいて複数の記号から新たな記号を生成

ルールによるサービス連携の表現:
Translator, MorphologicalAnalyzer, Dictionary
→ TranslationWithDictionary

- サービス提供者やアプリケーションのポリシーにより，あるサービス連携の構成が，他のサービス連携の可能性と同時に実現できない ⇔ 新たな信念が，すでに導かれた他の信念と矛盾

ポリシーによる衝突の例:
1. TranslationWithDictionary (**Translator**, MorphologicalAnalyzer, Dictionary), TranslationWithParallelText (**Translator**, **Parser**, ParallelText), SimilarityCalculator → **TranslationSelection**
2. **Translator, Parser → ⊥** (サービス提供者やアプリケーションのポリシー)
3. TranslationWithDictionary (Translator, MorphologicalAnalyzer, Dictionary), Translator, SimilarityCalculator → **TranslationSelection**

形式的定義

- ノード
  - 抽象/複合サービスに対応
  - < name, type, label, justification >
    - **name**: サービス名
    - **type**: サービスタイプ
    - **label**: サービスを実現しうる環境の集合
      (環境: サービスが実現されるのに必要な抽象サービスの集合)
    - **justification**: サービスを実現する他のサービスのノード群とのリンク
- 複合サービスモデル
  - $type_1$, $type_2$, ..., $type_n$ → (name, type)
- 矛盾
  - アプリケーション設計者やサービス提供者に与えられた制約
  - $name_1$, $name_2$, ..., $name_n$ → ⊥

形式的定義の例

- 抽象サービス
  - (*AbstractTranslator, MT, {{}}, {()}*)
  - (*AbstractDictionary, Dic, {{}}, {()}*)
- 複合サービスモデル
  - *MT, MA, Dic* → (*TransWithDic, MT*)
  - *MT, DP, PT* → (*TransWithPT, MT*)
  - *MT, MT, Sim* → (*TransSelect, MT*)
- 矛盾
  - *AbstractTranslator, AbstractParser* → ⊥

形式的定義の例

- *複合サービス*

(TransSelect,

MT,

Label

{{AbstractTranslator, AbstractSimilarityCalculator, AbstractDictionary, AbstractMophologialAnalyzer},
{AbstractTranslator, AbstractSimilarityCalculator, AbstractParallelText, AbstractParser},
{AbstractTranslator, AbstractSimilarityCalculator, AbstractDictionary, AbstractMophologialAnalyzer, AbstractParallelText, AbstractParser}},

Environments

{(AbstractTranslator, TransWithDic, AbstractSimilarityCalculator),
(AbstractTranslator, TransWithPT, AbstractSimilarityCalculator),
(TransWithDic, TransWithPT, AbstractSimilarityCalculator)}

Justification

形式的定義の例

- *複合サービス*

(TransSelect,

MT,

Label

{{AbstractTranslator, AbstractSimilarityCalculator, AbstractDictionary, AbstractMophologialAnalyzer},
{**AbstractTranslator**, AbstractSimilarityCalculator, AbstractParallelText, **AbstractParser**},
{**AbstractTranslator**, AbstractSimilarityCalculator, AbstractDictionary, AbstractMophologialAnalyzer, AbstractParallelText, **AbstractParser**}},

Environments

{(AbstractTranslator, TransWithDic, AbstractSimilarityCalculator),
(AbstractTranslator, TransWithPT, AbstractSimilarityCalculator),
(TransWithDic, TransWithPT, AbstractSimilarityCalculator)}

New justification

AbstractTranslator, AbstractParser → ⊥

形式的定義の例

- *複合サービス*

(TransSelect,

MT,

Label

{{AbstractTranslator, AbstractSimilarityCalculator, AbstractDictionary,
AbstractMophologialAnalyzer},
{~~AbstractTranslator, AbstractSimilarityCalculator, AbstractParallelText,~~
~~AbstractParser~~},
{~~AbstractTranslator, AbstractSimilarityCalculator, AbstractDictionary,~~
~~AbstractMophologialAnalyzer, AbstractParallelText,~~ ~~AbstractParser~~}},

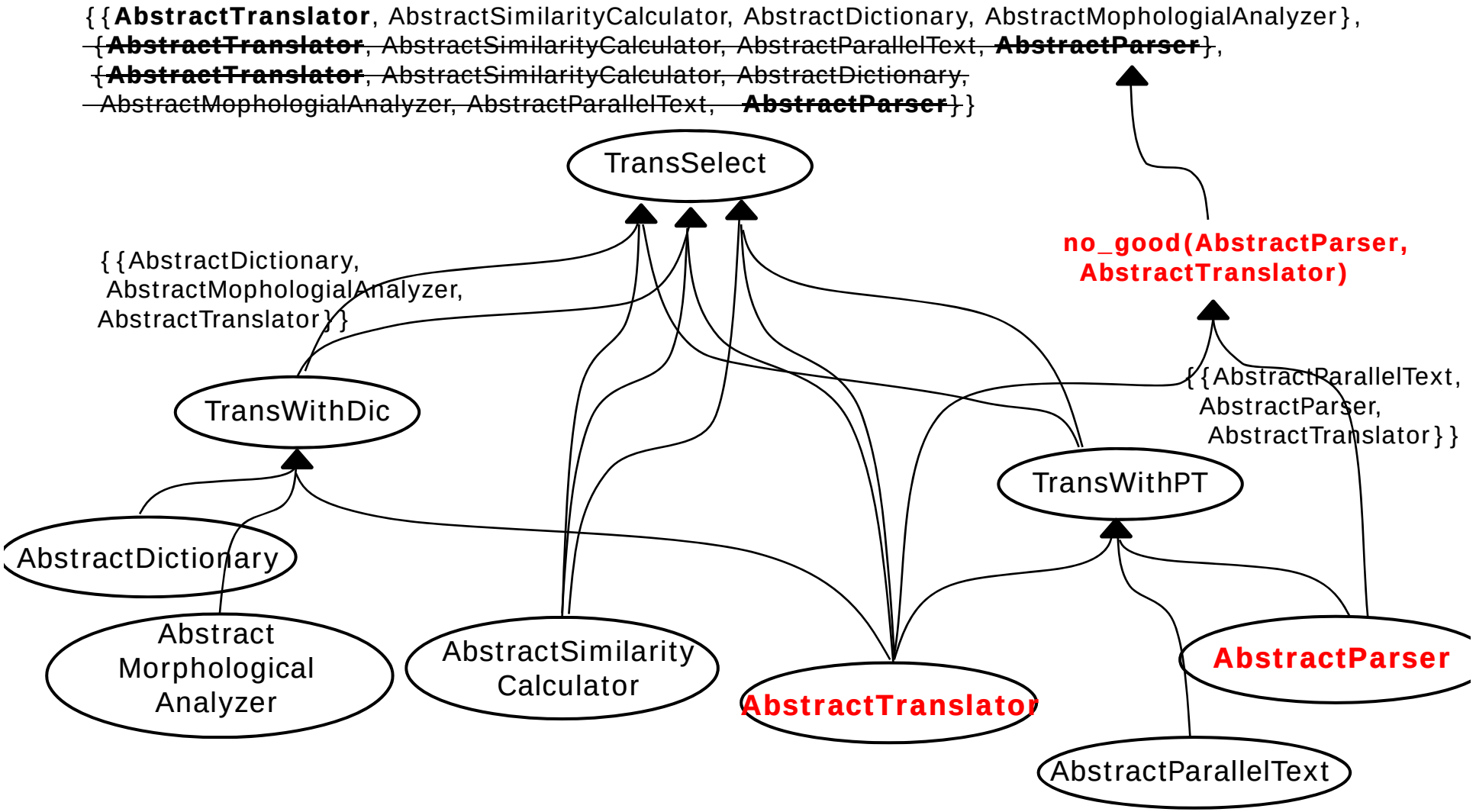
Environments

{(AbstractTranslator, TransWithDic, AbstractSimilarityCalculator),
(AbstractTranslator, TransWithPT, AbstractSimilarityCalculator),
(TransWithDic, TransWithPT, AbstractSimilarityCalculator)}

New justification

AbstractTranslator, AbstractParser → ⊥

バインディング依存グラフ



バインディング依存グラフの構築

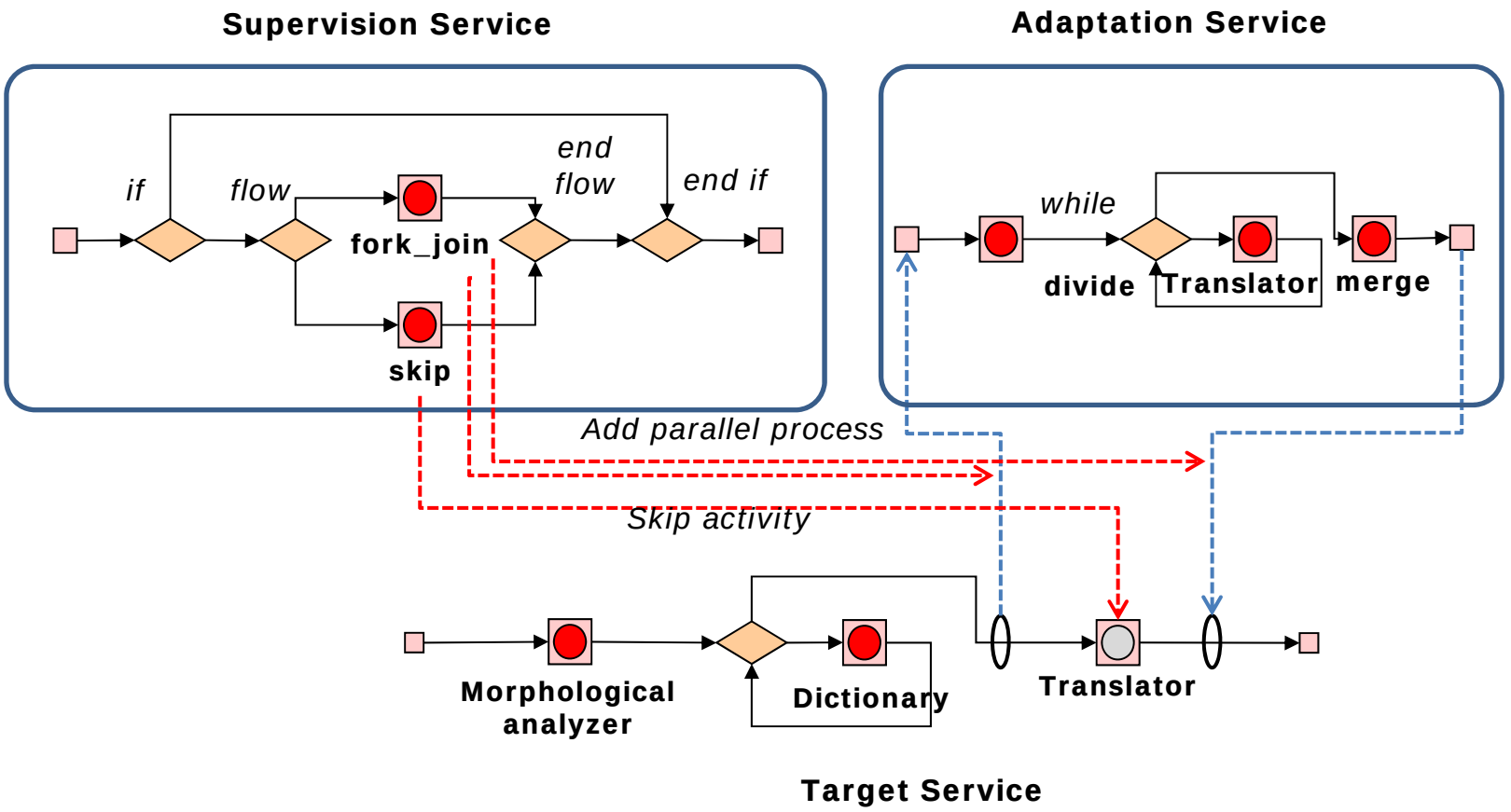
s ← *{si}*, *Sd* (ユーザが指定したサービス) を用いて以下を実行

1. 依存するノードのラベルに基づいて，ラベルを計算
   $$L_{new} = \{\cup\, e | e \in {}^{\forall} i \text{ (label of } s_i)\}$$
2. L’ ← すべてのnogoodとそれらを包含する環境を削除
3. L’’ ← *Sd* および同じラベルの他の環境を包含する環境を削除
4. if ラベルに変更がなければ終了
5. if *s* が矛盾だったら
   1. L’’のすべての環境をnogoodに
   2. すべてのノードのラベルからnogoodを削除
6. sに依存するノードのラベルに，ラベルの変更を伝播

有効な環境が存在すれば，それらを満たす具象サービスの組み合わせをCSPソルバによって探す

実行制御APIによる適応

- ビジネスロジックを実行時に変更
- 実行制御API（fork, join, skip, etc.）からなる複合サービスを並行して実行



バインディング依存グラフの更新

- 実行制御の適用時に，既存のグラフの変更
- CSPソルバを新たに得られたラベルの各環境について実行

1. cstarget‘ ← adapt(cstarget) // 複合サービスモデルを変更
2. *j* ← 推論エンジンで *cstarget‘* を使ったjustificationを計算
3. *L* ← *j*の帰結のノードのラベルを計算
4. CSPソルバで，*L* の各環境について具象サービスの割り当てを発見

実験

評価対象

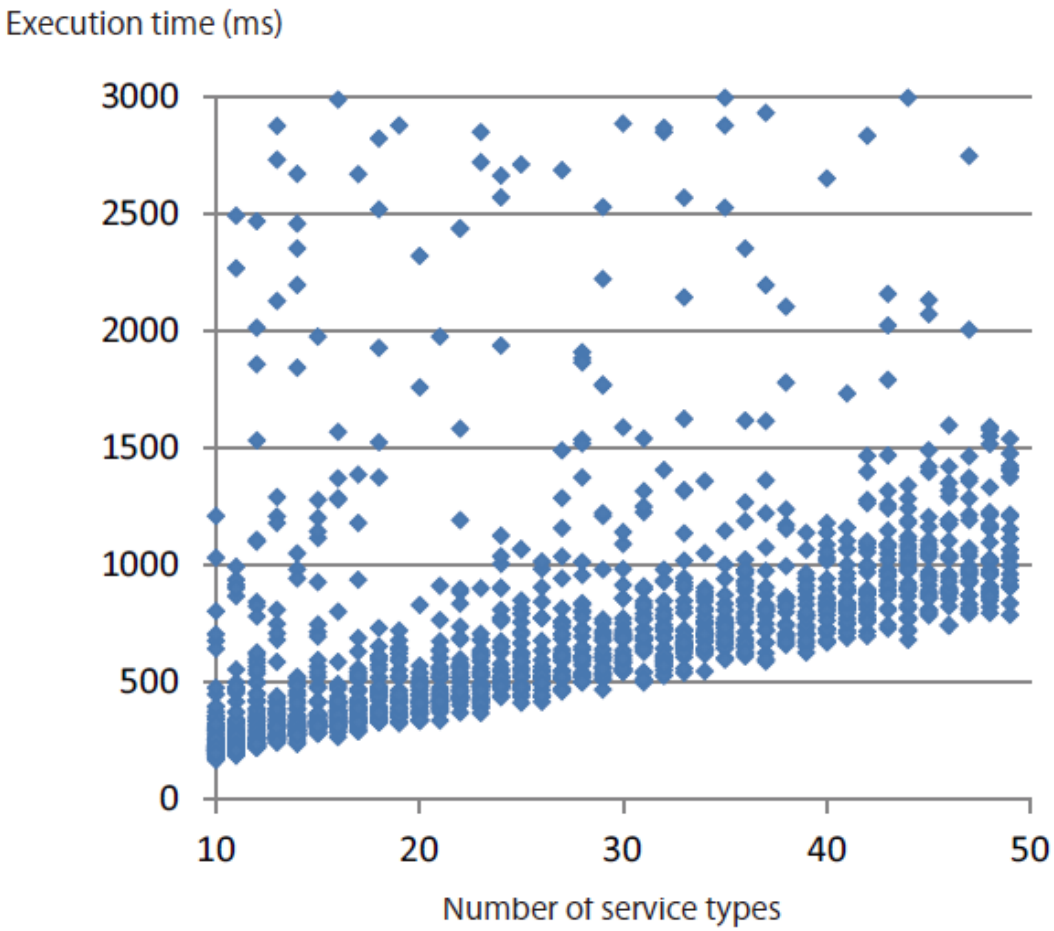
- バインディング依存グラフの構築時間
- バインディングの更新時間

規模の設定

- サービスタイプ数: ***n*** (10 ～ 50)
- 複合サービスモデル数: ***n***
  - 構成要素のサービス数はいずれも **3**
  - タイプはランダムに設定
- 各サービスクラスタの具象サービスの数: **20**
- 制約: 異なるタイプのサービス2つの組み合わせを禁止する**nC2 × *r*** の制約を設定
  - *r* は2つのサービスクラスタ間に定められた制約の数に相当
  - r= **10に設定**

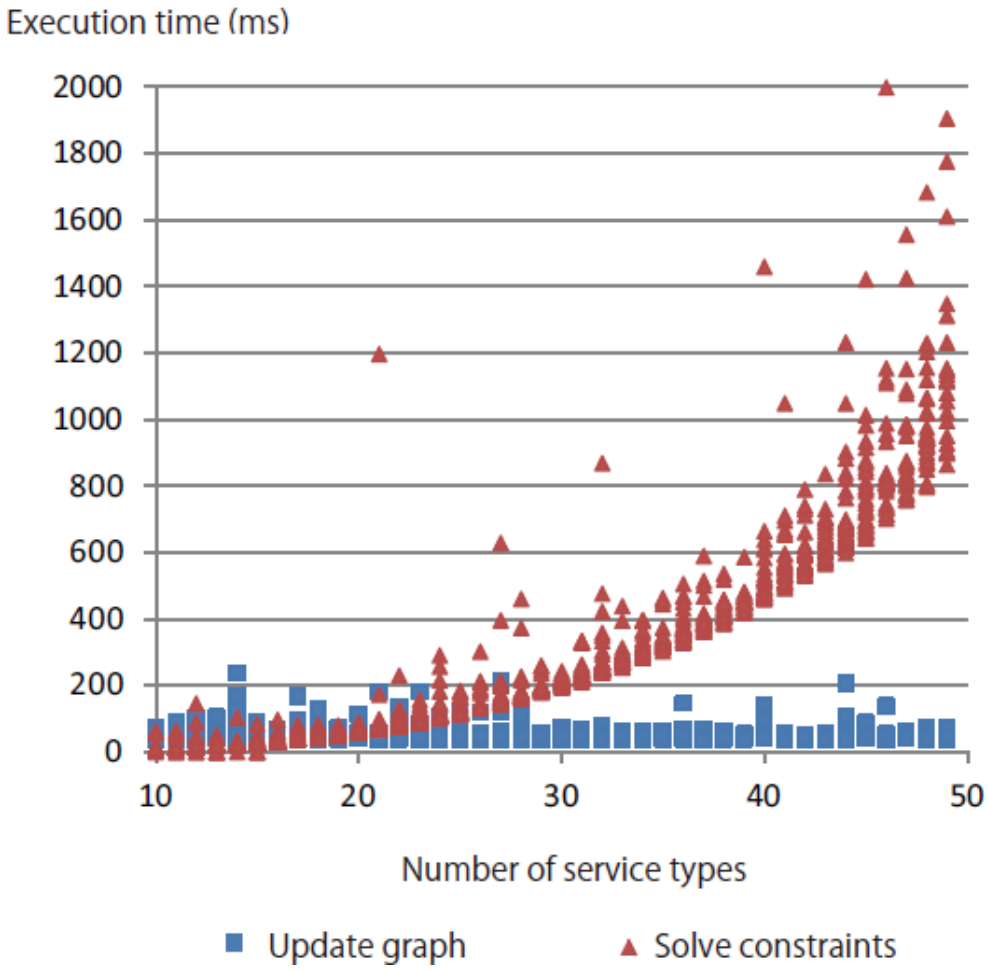
バインディング依存グラフの構築

- サービスタイプの数に対して，実行時間はある程度**線形**に増加
- ランダムに生成された設定によって，ラベルの更新にかかる**時間が大きくばらつく**



バインディングの更新

- グラフの更新は定数時間
  - ラベルの変更を伝播させないため
- 具象サービスの決定時間は指数的に増大:
  - CSPソルバの計算コストはサービスタイプ数(具象サービス数)に基づいて増大
  - 制約の密度が下がれば処理時間は短くなる



まとめ

- 研究の目的
  - オープン環境で階層的なサービス連携のバインディングを計算
  - 制約の動的な追加，実行時適応に対応
- アプローチ: ATMSのアイデアを適用
- 貢献
  - 制約の追加を効率的に扱えるバインディング依存グラフのためのモデルとアルゴリズムを提案
  - 実行時制御に応じたバインディング依存グラフの修正手法を提案