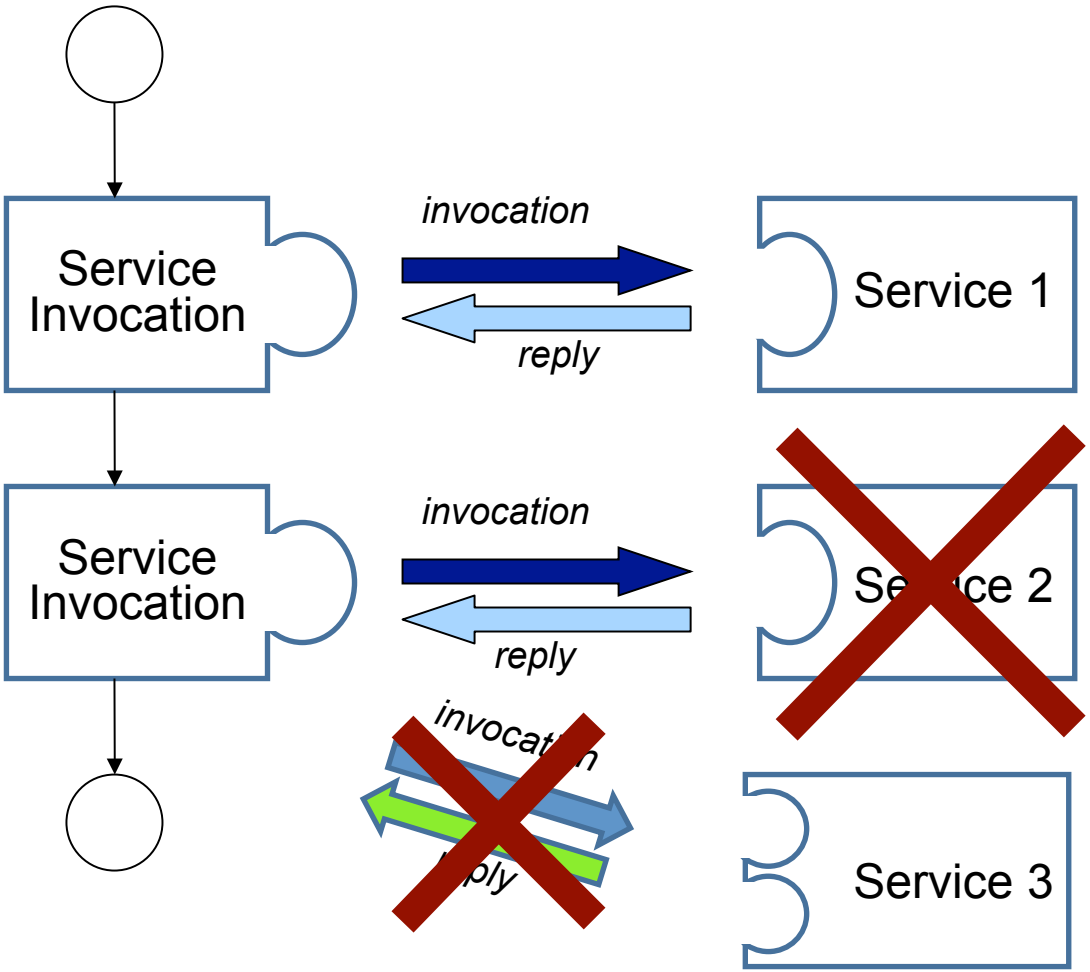
Synthesizing adapters for conversational web-services from their WSDL interface

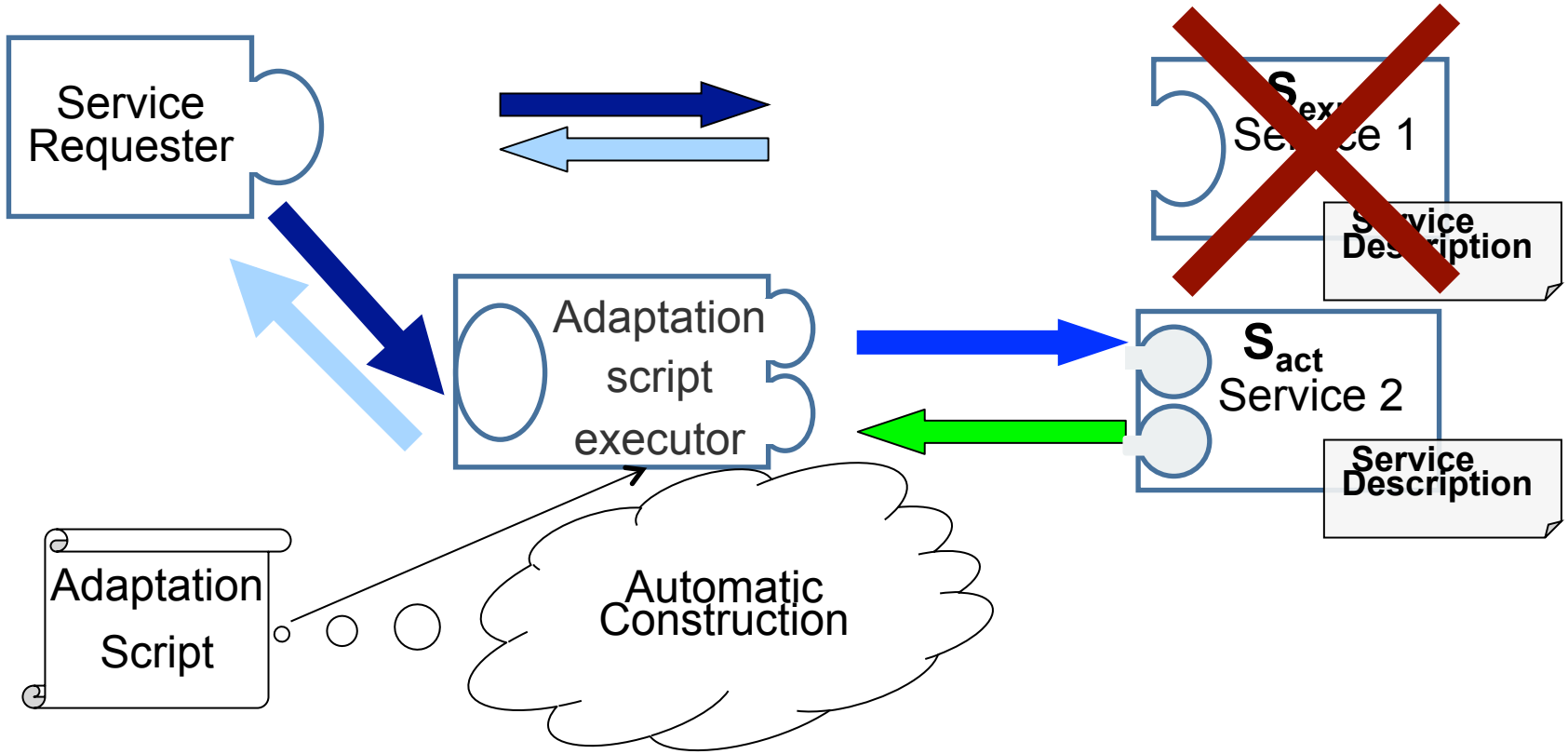
Luca Cavallaro, Elisabetta Di Nitto, Patrizio Pelliccione, Matteo Pradella, Massimo Tivoli

SOA and dynamic binding



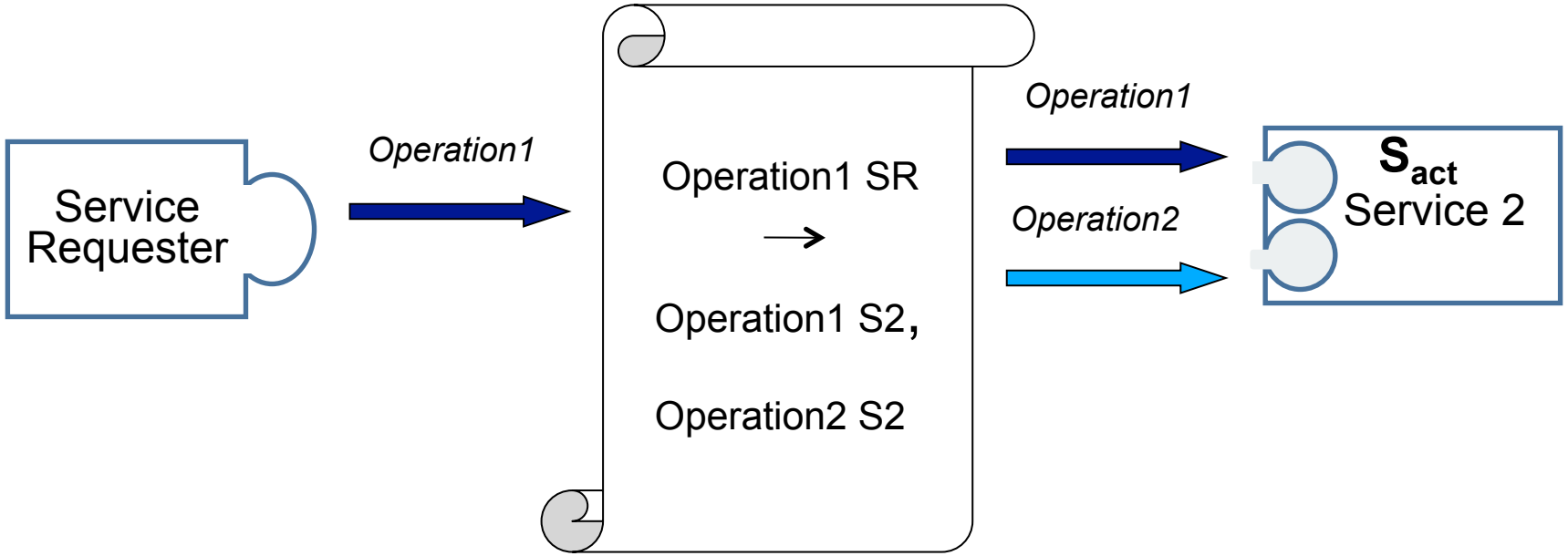
Adaptation Script

Our solution to the problem



Adaptation scripts

- Adaptation scripts define operation sequences of finite length



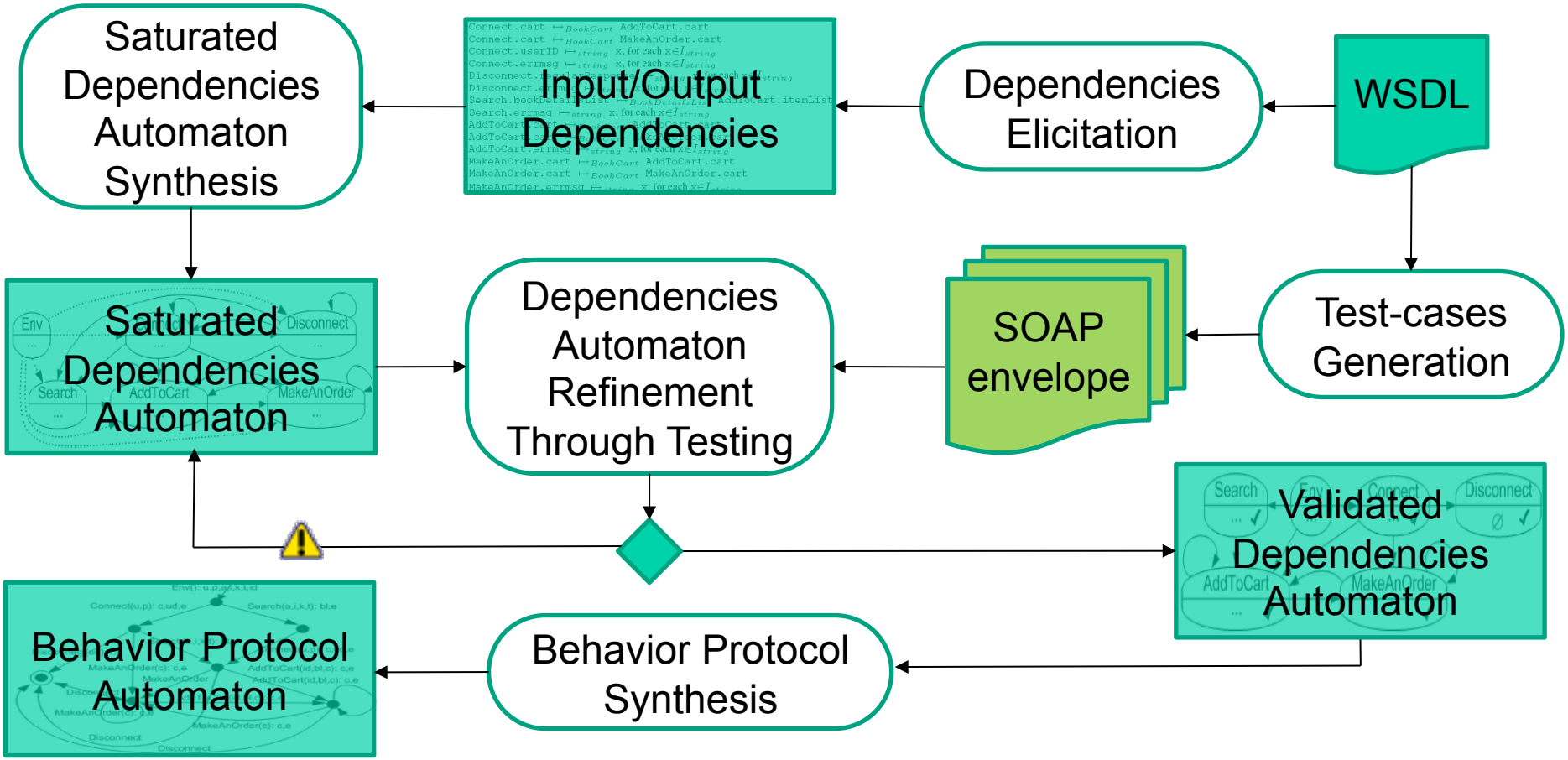
Service description

- Services are described in terms of:
  - Operations names and parameters
    - i.e., encoded in terms of WSDL or equivalent for REST
  - Order of operations invocation (protocol)
    - Defined through a *Behavior Protocol Automaton* (BPA)
- Usually only operation names and parameters are provided

Problem: need for a *rich* description

- Usually only operation names and parameters are provided
  - E.g. No BPA in WSDL
- *StrawBerry* discovers BPA from provided description
  - Derive BPA from a partial ordering relation among the possible invocations to the WSDL operations
  - Discovery process is based on testing, black-box and extra-procedural

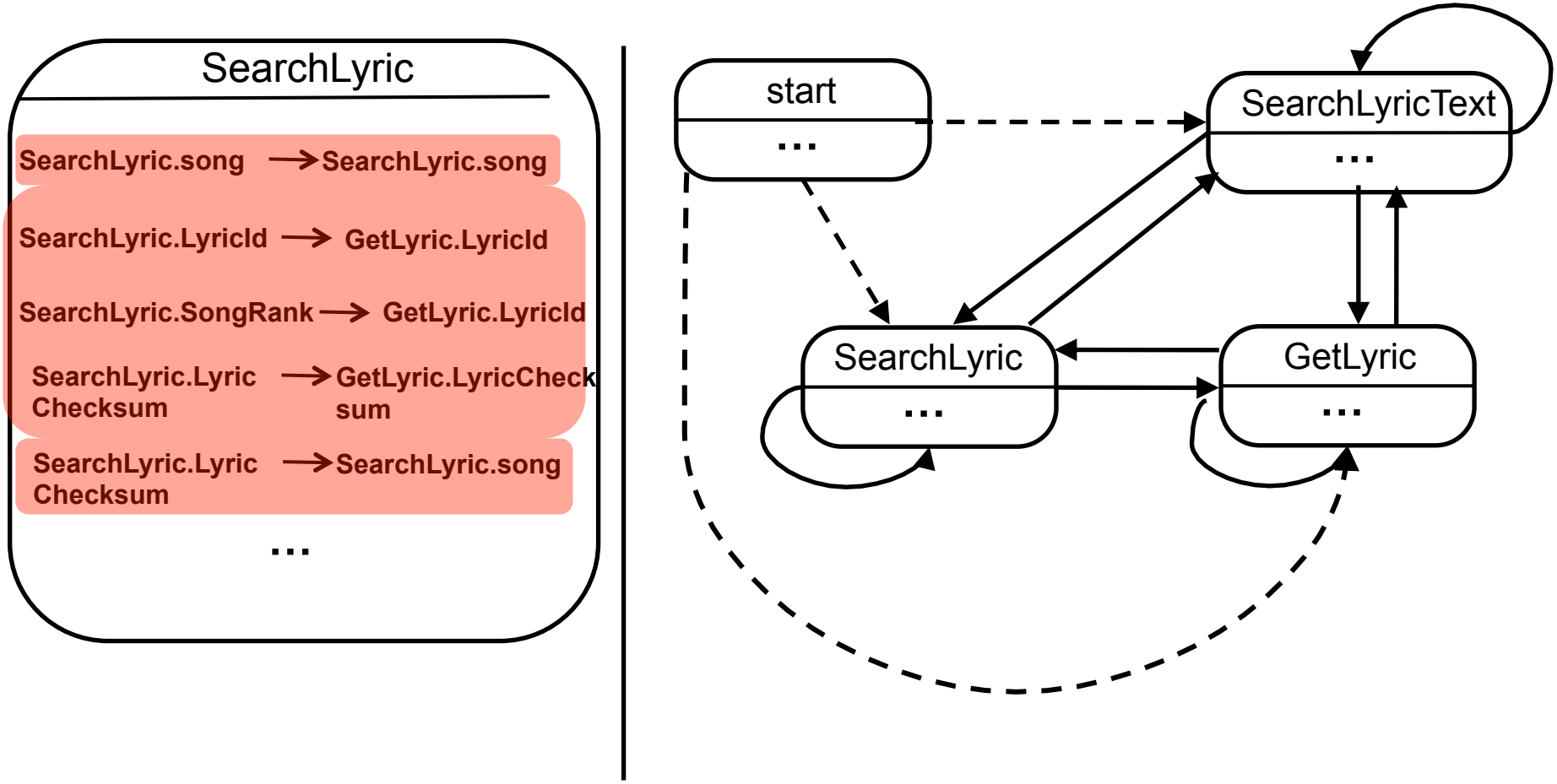
The StrawBerry method overview



Activity 1, dependency elicitation

Operation	Input data	Output data
SearchLiric	artist: string song: string	ArrayOf: LyricChecksum: string; LyricId: int; SongUrl: string; ArtistUrl: string; Artist: string; Song: string; SongRank: int;
SearchLiricText	lyricText: string	ArrayOf: LyricChecksum: string; LyricId: int; SongUrl: string; ArtistUrl: string; Artist: string; Song: string; SongRank: int;
GetLiric	LyricId: int; lyricCheckSum: string;	ArrayOf: LyricSong: string; LyricArtist: string; LyricUrl: string; LyricCovertArtUrl: string; LyricRank: int; Lyric: string;

Activity 2, dependencies automaton synthesis



Activity 3, test-cases generation

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.chartlyrics.com/">
  <soapenv:Header/>
  <soapenv:Body xmlns="http://api.chartlyrics.com/">
    <api:SearchLyric>
      <api:artist>KOVjotMZBEfbeynkhtAviBIEs</api:artist>
      <api:song>HJDSFGhfklskjgHFSKJnee</api:song>
    </api:SearchLyric>
  </soapenv:Body>
</soapenv:Envelope>
```

Activity 4, automaton refinement through testing

- The objective is to either prune false deps. or promote uncertain deps. to certain deps.
  - step1: positive tests in order to prune false deps.
  - step2: negative tests in order to confirm uncertain deps. as certain
  - step3: (experimentally few) remaining uncertain deps. are directly pruned

Activity 2, dependencies automaton synthesis

SearchLyric

SearchLyric.song → SearchLyric.song

SearchLyric.LyricId → GetLyric.LyricId ✓

SearchLyric.SongRank → GetLyric.LyricId

SearchLyric.Lyric Checksum → GetLyric.LyricCheck sum

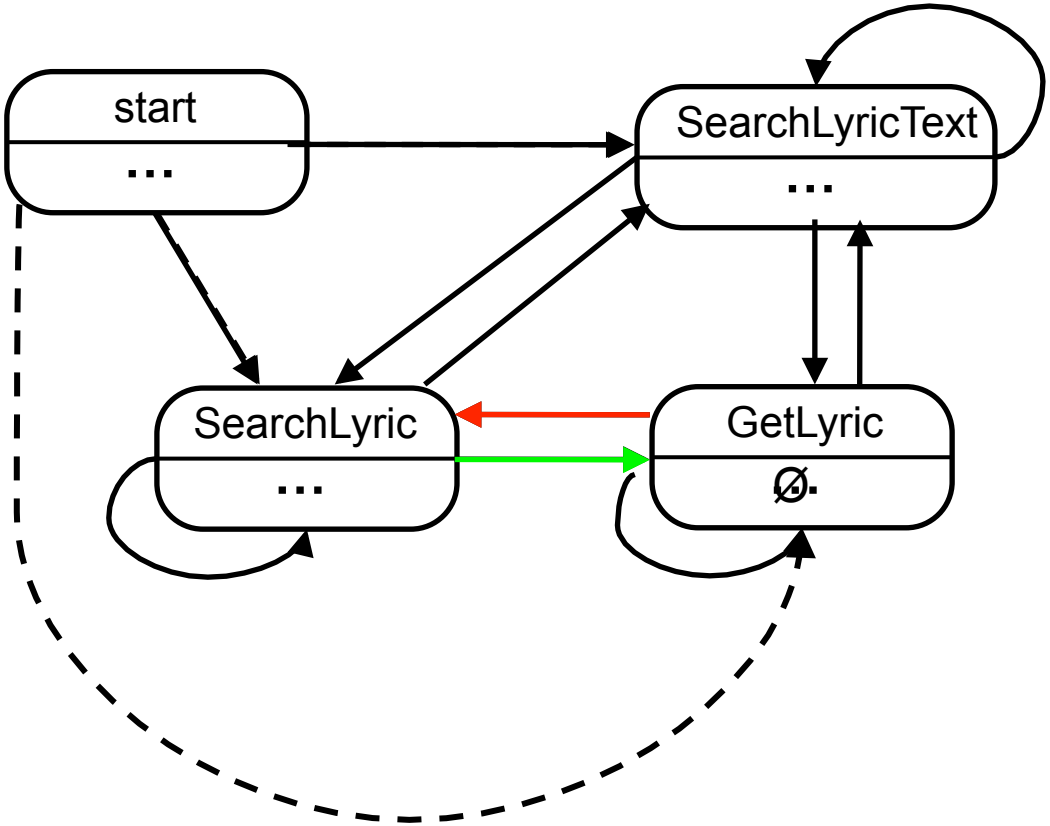
SearchLyric.Lyric Checksum → SearchLyric.song

...

GetLyric

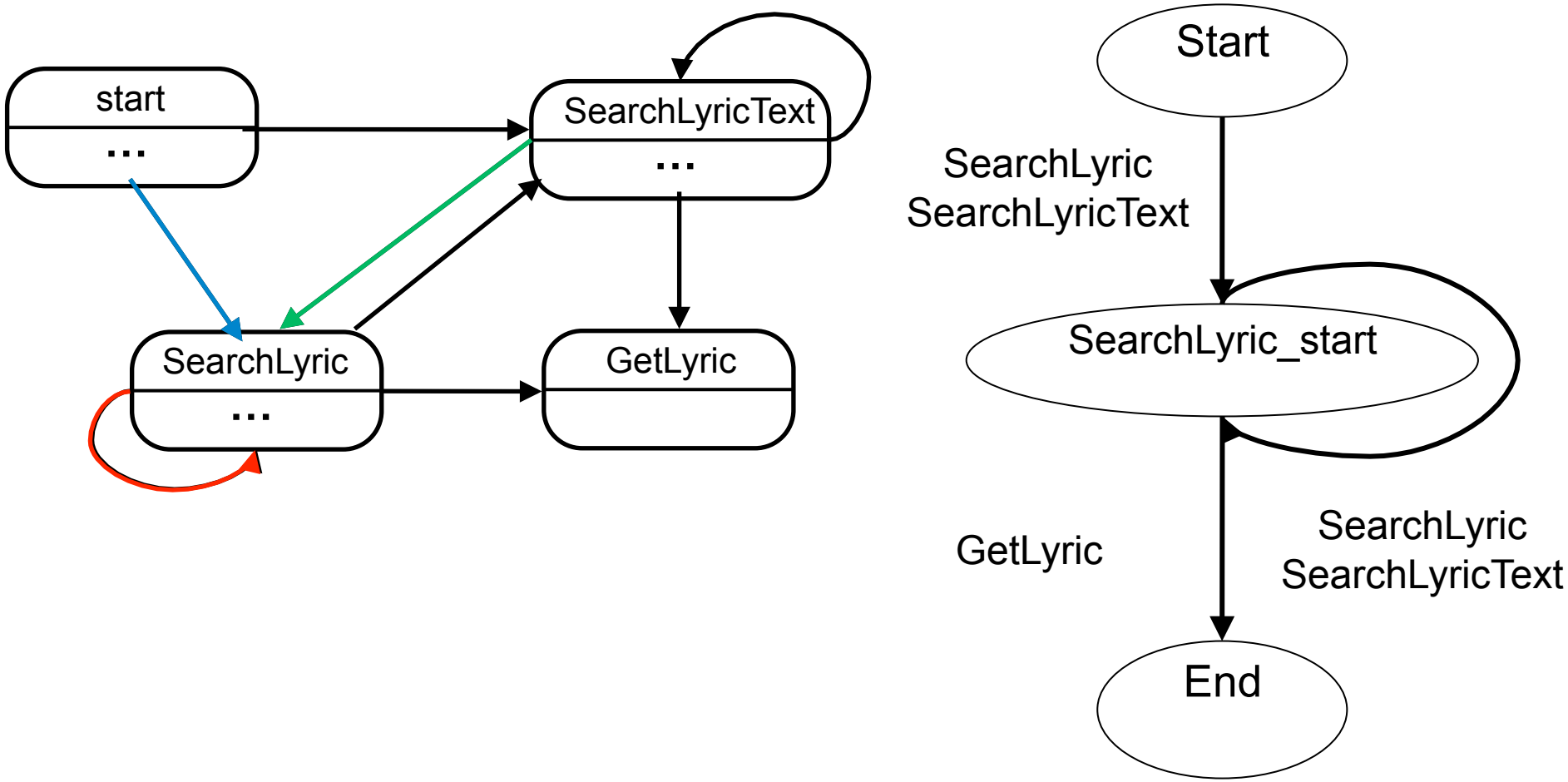
GetLyric.Lyric → SearchLyric.song

...

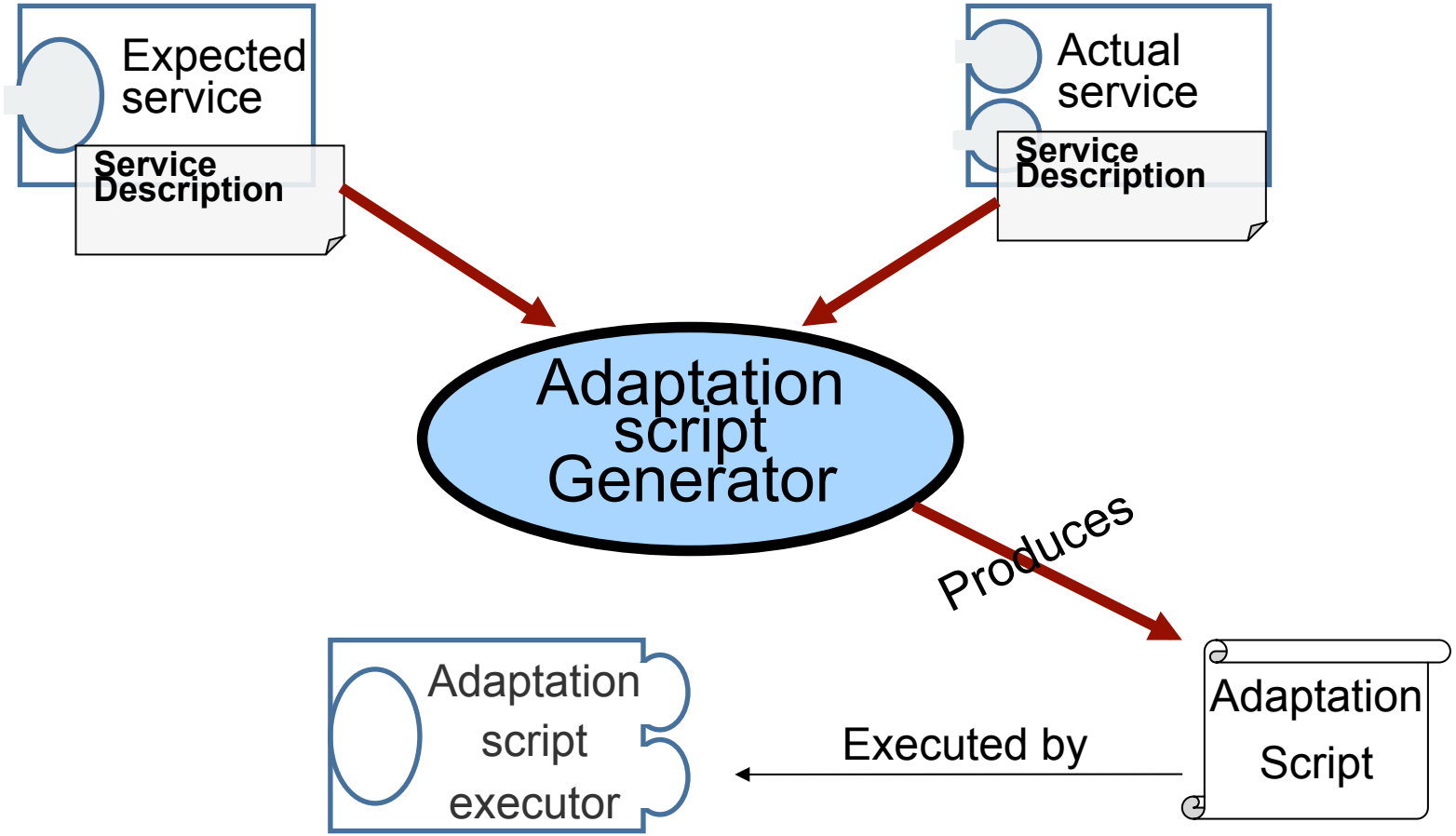


...

BAP synthesis



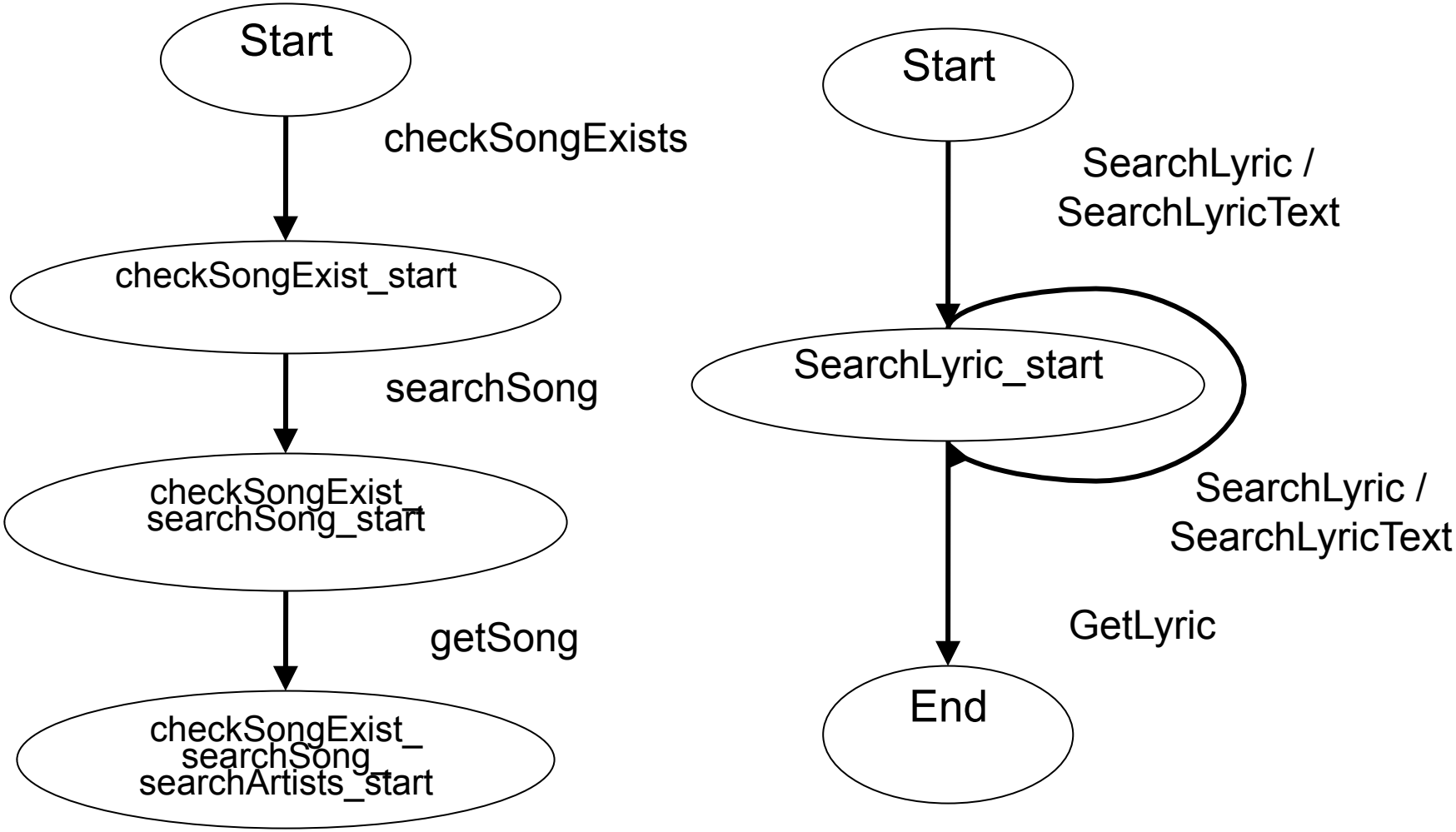
Automatic generation of adaptation scripts



Analysis of Expected and candidate Actual services

- Service descriptions analysis is performed by searching in S_{act} a sequence equivalent to that highlighted in S_{exp}
- A sequence in S_{act} is equivalent to a sequence in S_{exp} if:
  - Starts in a state s^i_{act} compatible with s^i_{exp}
  - Ends in a state s^f_{act} compatible with s^f_{exp}
  - Needs as input data a subset of those provided by $s^i_{act} \ldots s^f_{act}$
  - Returns as output parameters at least all the data needed by operations in $s^i_{act} \ldots s^f_{act}$

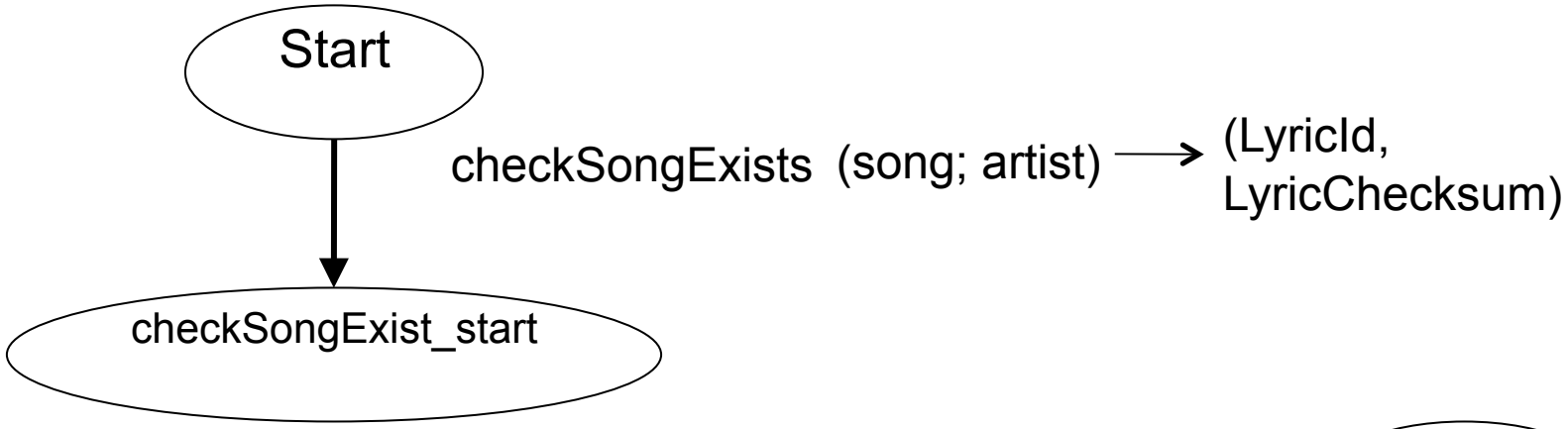
Service description analysis: Step 1

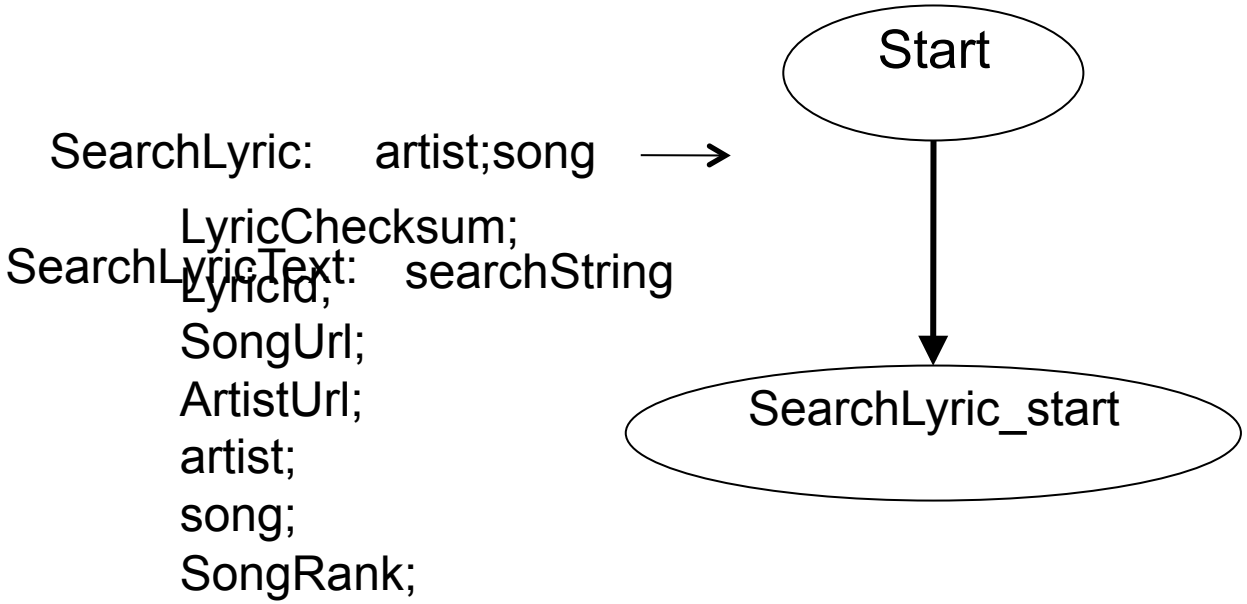


Service description analysis: Step 2

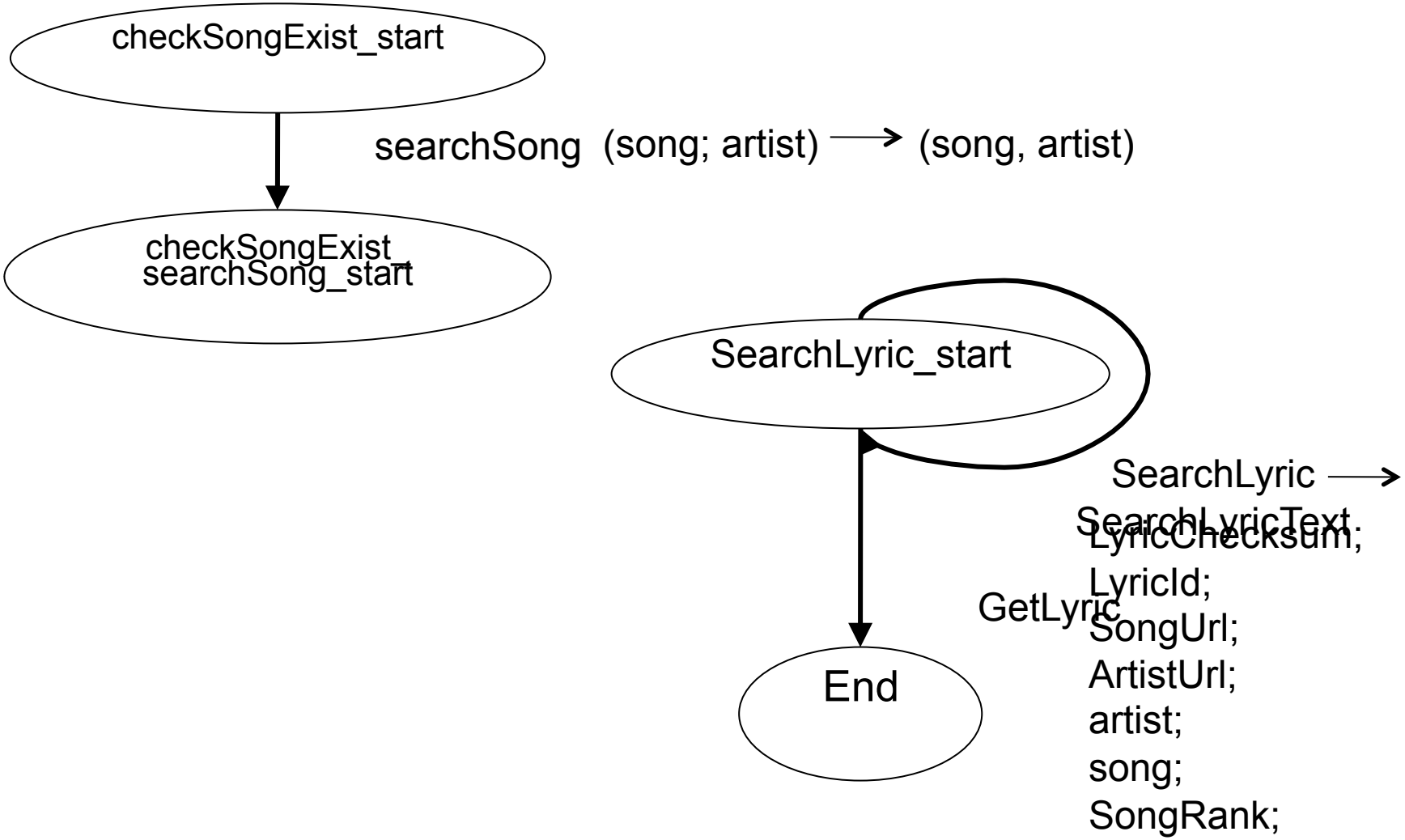
- Two possible search semantics:
  - Synchronous: if an operation in S_{exp} returns some data, no other operation in S_{exp} should be called until those data have been returned
  - Asynchronous: return data required by an operation in S_{exp} can be received also after subsequent operations have been invoked

Example

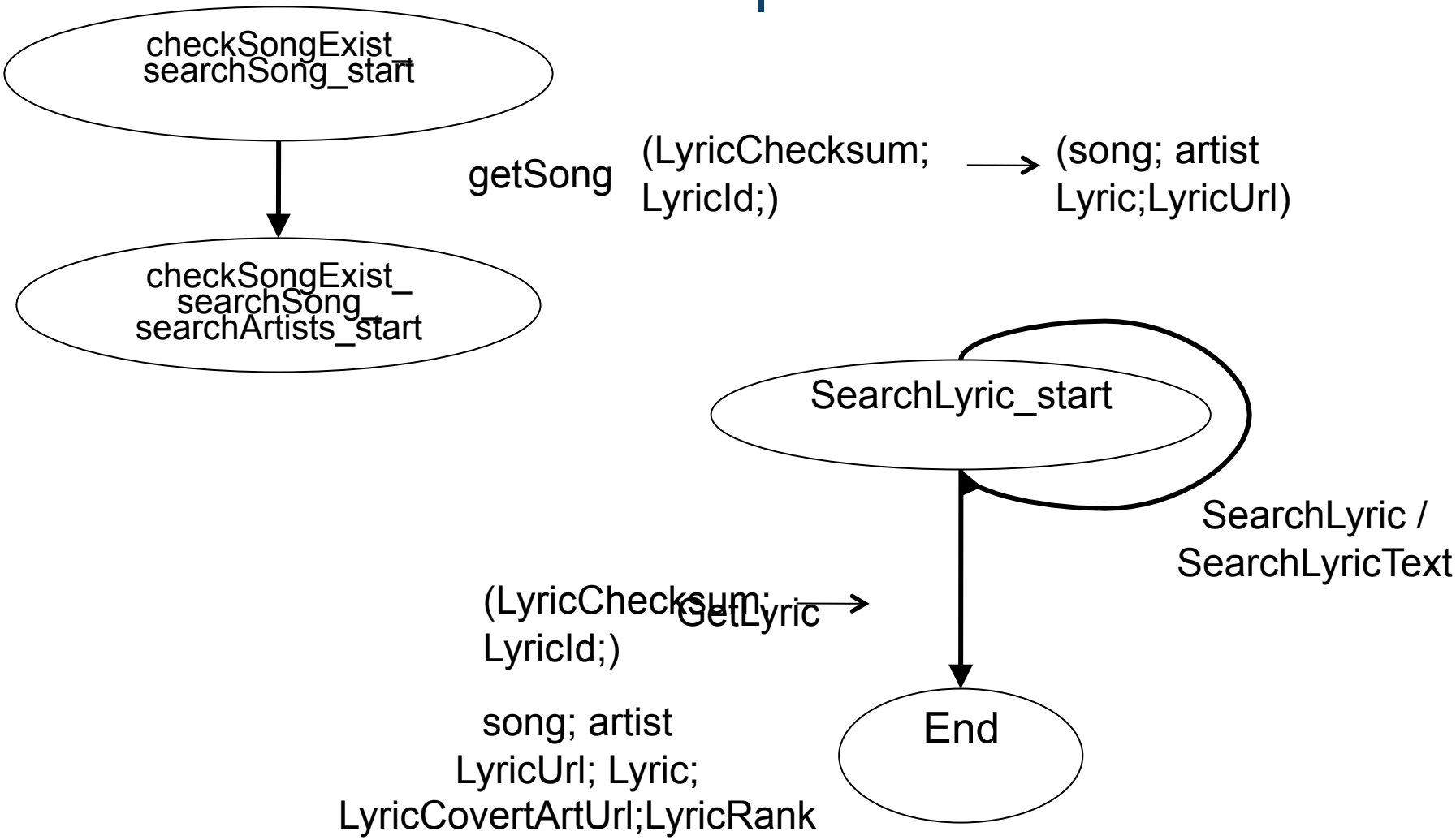

Start
checkSongExists (song; artist)
(LyricId,
LyricChecksum)
checkSongExist_start


Start
SearchLyric: artist;song
LyricChecksum;
SearchLyricText: LyricId; searchString
SongUrl;
ArtistUrl;
artist;
song;
SongRank;
SearchLyric_start

Example


checkSongExist_start
searchSong (song; artist) → (song, artist)
checkSongExist_
searchSong_start
SearchLyric_start
SearchLyric →
SearchLyricText;
LyricChecksum;
LyricId;
GetLyric
SongUrl;
ArtistUrl;
artist;
song;
SongRank;
End

Example


checkSongExist_
searchSong_start
getSong
(LyricChecksum;
LyricId;)
(song; artist
Lyric;LyricUrl)
checkSongExist_
searchSong_
searchArtists_start
SearchLyric_start
SearchLyric /
SearchLyricText
(LyricChecksum;
LyricId;)
GetLyric
song; artist
LyricUrl; Lyric;
LyricCovertArtUrl;LyricRank
End

Adaptation script generation in practice

- Mapping strategies are implemented by bounded model checking
- S_{exp} and S_{act} protocols are encoded in the input format of *Zot,* a SMT-solver based model checker
- The output is a finite length operations sequence which is used as an adaptation script

Conclusion

- Main contribution: A tool-supported methodology to enable service replacement
  - Input: WSDL of two services
  - Output: Adaptation script
- The methodology showed up to be feasible on the example
- Future work:
  - Validate Methodology
    - http://home.dei.polimi.it/cavallaro/sefm10-experiments.html
  - Improve protocol elicitation through heuristics