کارگاه‌های آموزشی مرکز اطلاعات علمی

آموزش نرم‌افزار Word برای پژوهشگران

روش تحقیق کمی

آموزش مهارت‌های کاربردی در تدوین و چاپ مقالات ISI

یک الگوریتم مبتنی بر گراف برای ترکیب سرویس ها

حسین قاسمعلی زاده
دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر
ghasemalizadeh@aut.ac.ir

محمدرضا رزازی
دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر
razzazi@aut.ac.ir

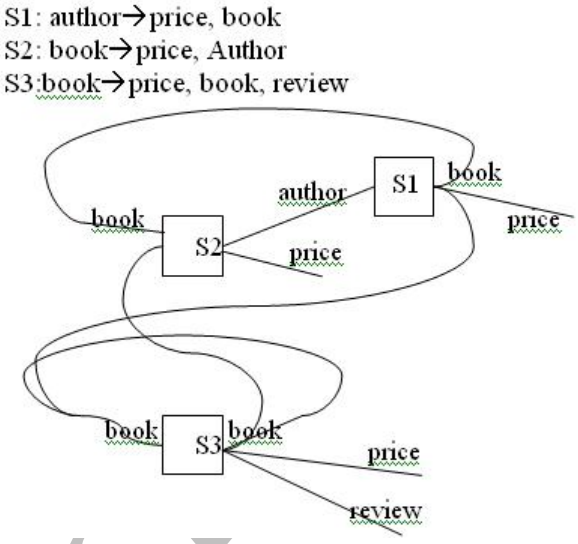
چکیده در این مقاله یک الگوریتم مبتنی بر گراف برای ترکیب سرویس ها ارائه شده است. اگر چه کارهای متفاوتی در رابطه با ترکیب سرویس ها موجود می باشد اما در این کارها تاکید بر جنبه های مختلفی از ترکیب سرویس ها مانند چگونگی بیان سرویس ها، چگونگی تطبیق میان سرویس ها، ارائه معماری برای ترکیب سرویس ها و... می باشد. در این مقاله تاکید ما بر خود الگوریتم ترکیب، درستی آن و تولید همه جواب ها می باشد. الگوریتم ارائه شده براساس درخواست کاربر به ساخت گراف ارتباطی میان سرویس ها می پردازد و با جستجو در گراف سرویس های ترکیبی مورد انتظار کاربر را پیدا می کند. گراف میان سرویس ها می تواند دارای گره ها و یال های فراوان و حلقه های متعدد باشد و نگهداری و بروز رسانی آن، بدلیل تغییرات در سرویس ها، مشکل است. الگوریتم ما گراف میان سرویس ها را به صورت یک گراف مستقیم بدون حلقه و متناسب با هر درخواست کاربر به گونه ای ایجاد می کند که تمامی سرویس‌های ترکیبی که می‌توانند پاسخ کاربر باشند را دارا باشد. پس از ایجاد این گراف مستقیم بدون حلقه ما با یک الگوریتم مسیریابی بازگشتی و با استفاده از یک پشته سرویس های ترکیبی را از آن استخراج می کنیم.

کلمات کلیدی: ترکیب سرویس‌ها، سرویس‌های معنایی، الگوریتم

۱- مقدمه

امروزه با استفاده از تکنولوژی وب سرویس می توان سرویس های مختلفی را در اینترنت و یا محیط های سرویس گرای دیگر به اشتراک گذاشت و کاربران می توانند متناسب با نیاز خود از این سرویس ها استفاده کنند. بطور کلی در هر محیط مبتنی بر سرویس اجزای ارائه دهنده سرویس، استفاده کننده سرویس و ذخیره گاه وجود دارد. ارائه دهندگان سرویس به معرفی سرویس های خود در ذخیره گاه می پردازند و استفاده کنندگان با استفاده از اطلاعات ذخیره شده در ذخیره گاه سرویس مورد نظر خود را پیدا می کنند و سرویس را مورد استفاده قرار می دهند. محیط مبتنی بر سرویس وظیفه دارد قابلیت های ارتباط میان ارائه دهندگان، استفاده کنندگان و ذخیره گاه را فراهم سازد. نمونه هایی از یک محیط مبتنی بر سرویس می تواند اینترنت، گرید و یا یک شبکه همتا به همتا باشد. با توجه به تعداد زیاد سرویس هایی که می توانند در این محیط ها وجود داشته باشد، برای استفاده کنندگان سرویس امکان شناسایی همه سرویس ها و استفاده از آن ها وجود ندارد. علاوه بر این ممکن است متناسب با درخواست کاربر یک سرویس خاص وجود نداشته باشد که خدمت مورد انتظار را ارائه دهد اما این امکان وجود داشته باشد تا از کنار هم قرار دادن سرویس های موجود و ترکیب آن ها درخواست کاربر را به انجام رساند. بدین ترتیب می توان با استفاده از سرویس های موجود درخواست های بیشتری از کاربران را بدون نیاز به اضافه کردن سرویس جدید برآورده کرد. به عنوان مثال چنانچه کاربر به سرویسی نیاز داشته باشدکه قیمت کتاب های یک نوسنده را برگرداند، می توان با ترکیب سرویسی که که کتاب های یک نویسنده را از نام نویسنده برمی-گرداند و سرویسی که با دریافت اطلاعات کتاب قیمت آن را برمی‌گرداند نیاز کاربر را پاسخ داد. هدف مساله ترکیب سرویس ها نیز همین موضوع یعنی ایجاد یک جریان کاری از سرویس ها برای پاسخ گویی به نیاز کاربر می باشد. به همین منظور، کاربر درخواست خود را با دادن ورودی و خروجی های مورد انتظار به سیستم می دهد. سیستم پس از دریافت ورودی ها و خروجی های مورد نظر کاربر، با جستجو در میان سرویس های موجود، سرویس ها و جریان های کاری از سرویس را که می توانند پاسخ گوی نیاز کاربر باشند به کاربر بر می گرداند.

سرویس هایی که در این سیستم مطرح است سرویس های ارائه دهنده اطلاعات می باشند. هر سرویس اطلاعاتی را به عنوان ورودی دریافت می کند و با انجام پردازش بر روی اطلاعات ورودی اطلاعاتی را به عنوان خروجی بر می گرداند. سرویس هایی که برای ترکیب در این مقاله در نظر گرفته شده اند سرویس های معنایی هستند. ما فرض می کنیم هر سرویس بصورت معنایی با استفاده از OWL-S[۱] شرح داده است. آنچه در این جا از شرح سرویس مد نظر است مفاهیم ورودی و خروجی سرویس می باشد که در شرح سرویس این مفاهیم آورده شده است. همچنین در شرح سرویس آنتالوژی که این مفاهیم در آن تعریف شده اند معرفی می شود. آن چه که از آنتالوژی در این مقاله اهمیت دارد روابط ارث بری میان مفهوم ها و ویژگی های هر مفهوم می‌باشد. فرض ما بر این است که آنتالوژی ها با استفاده از OWL[۲] شرح داده شده اند. از شرح سرویس و آنتالوژی مربوطه برای تعیین میزان شباهت میان ورودی و خروجی سرویس ها و این که آیا خروجی یک سرویس می تواند به عنوان



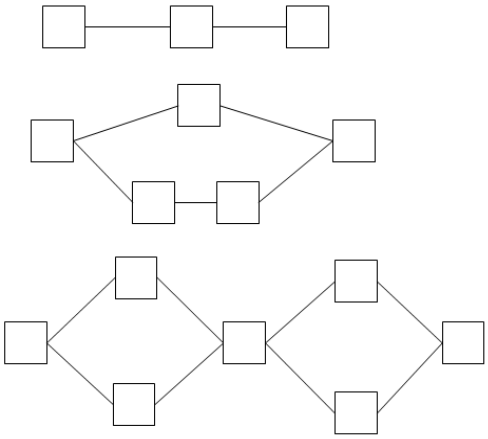
شکل ۱:نمونه ای از گراف میان سرویس ها

ورودی سرویس دیگر بکار رود استفاده می شود. از آن جایی که در این مقاله تمرکز ما بر رابطه شباهت نمی باشد، ما از رابطه شباهتی که در [۳] آمده استفاده می کنیم.

در ادامه در بخش ۲ کارهای مرتبط آورده شده است. در بخش ۳ راه حل ارائه شده معرفی شده است و در بخش ۴ الگوریتم ترکیب شرح داده شده است. اثبات الگوریتم در بخش ۵ آورده شده است.

۲- کارهای مرتبط

ایده کلی ترکیب سرویس ها با استفاده از ایجاد گراف میان سرویس ها تا کنون در چند کار [۸،۷،۶،۵] به عناوین مختلف مطرح شده است. در [۵] روش ارائه شده برای ترکیب سرویس ها با استفاده از گراف فرض می کند گراف سرویس ها بصورت کامل وجود دارد و همچنین فرض می کند مبدا و مقصد یک سرویس است و فقط یک مسیر را پیدا می کند. [۶] قصد ارائه یک چارچوب برای ترکیب معنایی سرویس‌ها دارد. در این چارچوب قالب هایی برای سرویس‌ها تعریف می شود و تاکید بر چگونگی انطباق میان قالب و سرویس های اصلی است. در روش ترکیب ارائه شده گراف میان سرویس ها برای کل سرویس ها نگهداری و بروز رسانی می شود و با استفاده از آن سرویس های ترکیبی پیدا می شوند. در این روش اشاره ای به حالاتی که ممکن است در گراف حلقه بوجود آید و یا برای یک درخواست چند پاسخ وجود داشته باشد نشده است. [۷] با ارائه یک ساختار معنایی برای بیان سرویس ها می‌پردازد و با استفاده از این ساختار معنایی یک معماری برای ترکیب سرویس ها ارائه می کند. در [۸]یک سیستم ترکیب سرویس مبتنی بر گراف برای سرویس هایی که یک ورودی و یک خروجی دارند ارائه شده است. در هیچ از یک کارهای ذکر



شکل ۲: نمونه هایی از ساختار سرویس های ترکیبی

شده جزییات ساخت گراف و حالت های مختلفی که ممکن است در استفاده از گراف پیش آید مورد بررسی قرار نگرفته است و تمرکز کار بر موارد دیگری بوده است. در این مقاله تمرکز ما بر چگونگی ساخت گراف و پوشش دادن حالت های مختلفی که در ساخت گراف ممکن است پیش آید می‌باشد. همچنین الگوریتم ما شرط خاصی را بر روی چگونگی بیان سرویس ها و یا تعداد ورودی و خروجی آن ها در نظر نمی‌گیرد.

۳- راه حل ارائه شده و نوآوری ها

الگوریتم ارایه شده در این مقاله مبتنی بر استفاده از گراف ارتباطی میان سرویس ها می باشد. منظور از گراف ارتباطی میان سرویس ها گرافی است که گره های آن سرویس ها و ورودی و خروجی آن‌ها می-باشند و یال های آن های نشان‌دهنده ارتباط میان ورودی‌ها و خروجی-های سرویس ها می باشند. این گراف در واقع دارای دو نوع گره می باشد. گره‌های از نوع سرویس که وابستگی میان ورودی‌ها و خروجی‌ها یک سرویس را نشان می دهند و گره های از نوع ورودی و خروجی (کلاس آنتالوژی) که برای نشان دادن ارتباط میان سرویس ها بکار می روند. یال های گراف در واقع یال هایی هستند که میان گره های از نوع ورودی و خروجی رسم می شوند. یک یال در صورتی به گراف اضافه می شود که خروجی یک سرویس بتواند به عنوان ورودی یک سرویس دیگر بکار رود. مشخص شدن اینکه آیا خروجی یک سرویس می تواند به عنوان ورودی سرویس دیگر بکار رود بر اساس رابطه شباهت میان کلاس های آنتالوژی، که ورودی و خروجی سرویس ها می باشند، تعیین می شود. به این ترتیب که اگر شباهت میان دو کلاس، که بر اساس رابطه شباهت بدست می آید، از میزان مشخصی بیشتر باشد، می توان دو سرویسی که خروجی و ورودی آن‌ها کلاس های مزبور می باشد را به هم متصل کرد.

```
FindServices {
  inputs = User Inputs;
  outputs = User Outputs;
  I = inputs;
  While (I != null){
    IS = {is | ∃in ∈ is.inputs & ∃out ∈ I & related(in, out)}
  O = O ∪ I;
    AppropriateIS = {is | is ∈ IS ∀in ∈ is.inputs in ∈ O};
  I = ConstructGraph(AppropriateIS, O);
  }// End While
  CheckGrpah();
  }// End
```

شکل۳: شبه کد الگوریتم ترکیب

گراف ارتباطی میان سرویس ها می تواند گراف پیچیده ای باشد. به عنوان مثال در شکل ۱ گراف ارتباطی برای سه سرویس نمونه، که از مجموعه OWL-TC[۴] انتخاب شده است، رسم شده است. همانطور که در این شکل مشاهده می شود، گراف ارتباطی میان سرویس ها می تواند دارای حلقه های متعددی باشد که این موضوع ساخت گراف و مسیریابی در آن پیچیده می نماید. از سوی دیگر وجود دو نوع گره در گراف و دو نوع وابستگی میان گره‌ها گراف سرویس ها را متفاوت از گراف های معمولی می سازد و الزامات خاصی را به ساخت و مسیریابی در گراف می-افزاید. بر همین اساس در این مقاله یک الگوریتم مبتنی بر گراف که بر اساس درخواست کاربر و بصورت افزایشی به ساخت گراف ارتباطی بصورت DAG[1] می پردازد، و پس از ساخت گراف نتایج را از آن استخراج می کند ارائه شده است. سرویس های ترکیبی که می تواند به عنوان پاسخ کاربر بکار روند می تواند در شکل های متفاوتی از لحاظ ارتباط بین سرویس ها ظاهر شوند. یک سرویس ترکیبی می تواند شامل اجرای سری چند سرویس، اجرای موازی چند سرویس و یا ترکیبی از اجرای سری و موازی باشد. شکل۲ نمونه های مختلفی از ساختار سرویس های ترکیبی را نشان می دهد..

برای این که در ساخت گراف بتوانیم حالت های مختلف ترکیب سرویس ها را پوشش دهیم ما نوآوری های زیر را بکار برده ایم:

- ساخت گراف ارتباطی میان سرویس ها بر اساس هر درخواست کاربر
- تبدیل گراف میان سرویسی به DAG، با جداسازی مفهوم سرویسی که در گراف قرار می گیرد از سرویس واقعی
- هرس کردن گراف با تعریف مفهوم سرویس مناسب.
- نوعی مسیریابی خاص در گراف که تمامی سرویس های ترکیبی را بر می گرداند.
- اثبات درستی الگوریتم

در ادامه مقاله الگوریتم را شرح می دهیم و اثبات آن را بیان می کنیم.

۴- الگوریتم ترکیب

شبه کد الگوریتم در شکل ۳ آمده است. الگوریتم با دریافت شرح سرویس مورد انتظار کاربر کار خود را شروع می کند. کاربر سرویس مورد انتظار خود را با دادن ورودی های سرویس و خروجی هایی که باید تولید کند، بیان می‌کند. در مرحله اول ورودی های داده شده توسط کاربر به کلاس هایی از در آنتالوژی نگاشت می شوند. به عنوان مثال چنانچه کاربر کلمه "book" را به عنوان ورودی سرویس مورد درخواست خود به سیستم بدهد، کلیه کلاس هایی که در آنتالوژی های مختلف به نام "book" می باشند به عنوان ورودی درخواستی کاربر در نظر گرفته می شوند. به همین ترتیب خروجی درخواستی کاربر به کلاس هایی از آنتالوژی های مختلف نگاشت می شود. روند کلی الگوریتم به این صورت است که در هر مرحله مجموعه ای از سرویسها را بازیابی می کند، سرویس های مناسب را از میان آن ها انتخاب می کند و با استفاده از آن ها گراف را بروز رسانی می کند. این کار تا زمانی ادامه پیدا کند که کلاس هایی برای برای اضافه کردن سرویس های جدید وجود داشته باشد. مجموعه I شامل مجموعه ای از کلاس های آنتالوژی می باشد که در هر تکرار از الگوریتم سرویس هایی که یکی از ورودی های آن عضوی از I باشد برای اضافه شدن به گراف انتخاب می شوند. در شروع کار I شامل ورودی های کاربر می باشد. مجموعه O شامل ورودی های کاربر و خروجی های کلیه سرویس هایی می باشد که تاکنون به گراف اضافه شده اند. اعضای O می توانند به عنوان ورودی سرویس های جدیدی که می توانند به گراف اضافه شوند بکار روند. در هر مرحله سرویس هایی قابلیت اضافه شدن به گراف را دارند که تمامی ورودی های آنها از ورودی های کاربر و خروجی های سرویس های موجود در گراف قابل فراخوانی باشند.

[1] Direct Acyclic Graph

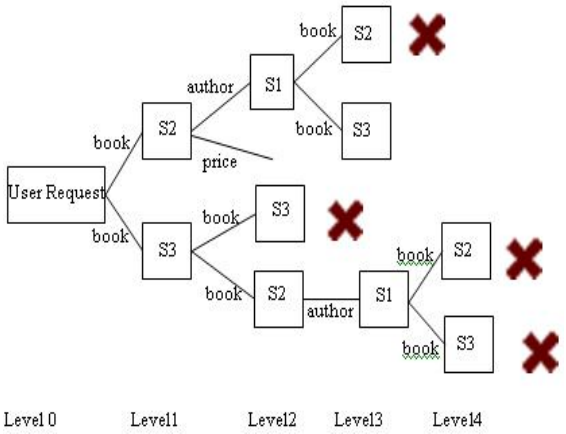
```
ConcstructGraph(Services,O){
 for each newSrv ∈ Services
  for each input ∈ newSrv.inputs
   for each o ∈ O
    if( realated(input,o))
     if( nochain(srv, o.getService()) &noRepeat()){
      previousSrv = o.getService();
      if(there is no serviceVertex  for newSrv at this level){
      Create a new serviceVertex for srv and
      connect it  to previosSrv; }
 if(there is one or more connection via input at this level) {
    Create a new serviceVertex for srv and connect
    it to previosSrv;
    add all previous connection via other
    newSrv inputs to this new serviceVertex; }
  if(there is no connection via input at this level) {
    update all serviceVertexes which was created
    for newSrv at this level with this new connection; }
  }
```

شکل۴: شبه کد ساخت گراف

ما این سرویس ها را سرویس های مناسب می نامیم. در واقع یک سرویس مناسب سرویسی است که کلیه ورودی های آن عضوی از مجموعه O هستند. مجموعه سرویس های مناسب را AppropriateIS می نامیم. در الگوریتم ترکیب پس از انتخاب سرویس های مناسب ساخت گراف صورت می پذیرد.

۴-۱ ساخت گراف

در این الگوریتم پس از انتخاب سرویس های مناسب مرحله ساخت گراف می باشد. ورودی این مرحله خروجی های سرویس های مناسب مراحل قبل O و سرویس های مناسب جدید *AppropriateIS* می باشد. اولین گره در گراف سرویسی مجازی است که خروجی های آن ورودی های کاربر می باشد و این سرویس سرویس مناسب مرحله صفر در نظر گرفته می شود که برای شروع ساخت گراف مورد استفاده قرار می گیرد. ساخت و نگهداری گراف از طریق نگهداری روابط میان سرویس ها صورت می گیرد. به همین منظور، مفهوم جدیدی را تحت عنوان ServiceVertex معرفی می کنیم(گره سرویس) که در واقع سرویسی



شکل۵: نمونه ای از DAG ایجاد شده توسط الگوریتم

است که در ساخت گراف مورد استفاده قرار می گیرد. هر گره سرویس شامل یک سرویس یک مجموعه سرویس های بعدی در گراف و یک مجموعه سرویس های قبلی می باشد. با جدا شدن مفهوم سرویس و گره سرویس یک سرویس می تواند چندین بار در گراف ظاهر شود و در ترکیب های مختلفی شرکت کند. استفاده از این مفهوم با عث می شود که گراف میان سرویس ها به DAG تبدیل شود و بتوان کلیه سرویس‌های ترکیبی ممکن را ایجاد کرد. ورودی تابع ساخت گراف کلیه خروجی های سرویس های مناسبی است که در مراحل قبلی به گراف اضافه شده اند (مجموعه O) به علاوه سرویس های مناسب جدید. ساخت گراف بدین گونه صورت می گیرد که هر یک از ورودی های سرویس های مناسب جدید با کلیه اجزای مجموعه O چک می شوند و در صورتی که از لحاظ رابطه آنتالوژی قابل اتصال باشند یک گره سرویس برای سرویس مناسب مورد بررسی ایجاد می گردد. در این میان چنانچه پارامترهای قبلی سرویس مورد بررسی اتصالاتی برقرار کرده باشند کلیه اتصالات قبلی به گره سرویس جدید اضافه می گردد. علاوه بر این اتصالی میان گره سرویسی که عضو مجموعه O خروجی آن می باشد و گره سرویس جدید برقرار می گردد. در این میان کلیه ارتباطات قبلی با نود سرویس جدید برقرار می شود و ارتباط جدید به گره سرویس های قبلی که از سرویس مناسب مورد بررسی ایجاد شده اند اضافه می شود. در هنگام ساخت گراف دو مورد دیگر بررسی می شود که عبارتند از تشکیل حلقه و تکرار. تشکیل حلقه به این معنا است که سرویسی که در یک مسیر آمده است، بار دیگر در مسیر قرار نگیرد و مورد گسترش واقع نشود. تکرار به این معنی است که یک سرویس تنها یک بار پس از سرویس دیگر قرار گیرد. در ساخت گراف باید از تشکیل حلقه و تکرار جلوگیری شود زیرا این کار باعث بوجود آمدن مسیر های نامعتبر و تکراری در گراف می شود. خروجی الگوریتم

```
FindResult(ServiceVertex service) {
  path.push(service);
  if (service.outputs().satisfy(UserOutputs)
      report path as one of results;
  foreach( newService ∈ service.next() )
      FindResults(newService);
  path.pop();
}
```

شکل۶: شبه کد پیدا کردن نتایج

ساخت گراف مجموعه کلاس هایی است که در تکرار بعدی برای آوردن مجموعه سرویس های جدید مورد استفاده قرار می گیرند. این مجموعه خروجی گره سرویس هایی می باشد که به گراف اضافه شده اند. چنانچه در یک تکرار گره سرویس جدیدی به گراف اضافه نشود خروجی الگوریتم ساخت گراف تهی خواهد بود که منجر به خاتمه الگوریتم خواهد شد. شکل ۵ نمونه ای از DAG ایجاد شده توسط الگوریتم ساخت گراف که برای سه سرویس شکل۱ ایجاد شده است، نشان می دهد. پس از ساخت گراف مرحله پیدا کردن نتایج صورت می‌پذیرد.

۴-۲ پیدا کردن نتایج

در این مرحله سرویس های ترکیبی مورد انتظار کاربر کاربر از گراف تشکیل شده در مرحله قبل استخراج می شوند. پیدا کردن نتایج در گراف با شروع از گره مجازی کاربر آغاز می شود و با دنبال کردن اتصالات ایجاد شده به صورت جستجوی اول عمق ادامه می یابد. الگوریتم پیدا کردن نتایج به صورت بازگشتی عمل می کند. در هر تکرار یک گره سرویس به عنوان ورودی به الگوریتم داده می شود. از آن جایی که یک سرویس ترکیبی از چند سرویس تشکیل شده است برای نگهداری مسیر در بین فراخوانی های متفاوت تابع پیدا کردن نتایج از یک پشته استفاده می شود. الگوریتم در ابتدا به بررسی سرویس ورودی می پردازد و بررسی می کند آیا این سرویس می تواند خروجی مورد انتظار کاربر را تولید کند. در صورتی که سرویس مورد بررسی بتواند خروجی های مورد انتظار کاربر را تولید کند، سرویس های موجود در پشته به عنوان یکی از نتایج برگردانده می شوند. پس از این مرحله هر یک از سرویس هایی که در گراف بعد از سرویس مورد بررسی قرار دارند، بصورت بازگشتی فراخوانی می شوند. پس از اتمام این فراخوانی ها سرویس مورد بررسی از پشته خارج می شود. شکل ۶ شبه کد بخش پیدا کردن نتایج را نشان می‌دهد.

۵- اثبات درستی

در این قسمت به اثبات درستی الگوریتم ارائه شده می پردازیم.

قضیه: الگوریتم تمامی سرویس های ترکیبی را که می توانند با ورودی های کاربر فراخوانی شوند و خروجی کاربر را تولید کنند، بر می گرداند.

تعریف: سرویس های ترکیبی مورد نظر سرویس هایی هستند که تمام ورودی های سرویس های عضو آن ها از ورودی های کاربر و خروجی های سرویس های قبلی در سرویس ترکیبی قابل فراهم شدن باشد و یک سرویس حداکثر یک بار در سرویس ترکیبی حاضر باشد.

لم ۱: گراف ایجاد شده از سرویس ها کلیه سرویس های ترکیبی را که می توانند با ورودی های کاربر فراخوانی شوند دارا می باشد.

اثبات با استفاده از ثابت حلقه: ثابت حلقه را این مورد در نظر می گیریم که در هر سطح از تولید گراف الگوریتم کلیه سرویس های ترکیبی را که می‌توانند با ورودی‌های کاربر فراخوانی شوند دارا می باشد.

در سطح ۱ گراف کلیه سرویس هایی که می توانند در گراف قرار داده شوند، وارد گراف شده اند. زیرا در این سطح کلیه سرویس هایی که ورودی های آن ها از ورودی های کاربر تامین می شود به گراف اضافه شده اند.

فرض می کنیم تا مرحله (I-1) ام تمام سرویس هایی که می توانند با یک جریان کاری با طول حداکثر I-1 که از ورودی های کاربر آغاز می شود، فراخوانی شوند به گراف اضافه شده اند. ثابت می کنیم در مرحله I ام تمام سرویس هایی که می توانند با یک جریان کاری به طول I که با ورودی های کاربر آغاز می شود فراخوانی شوند، به گراف اضافه می شوند.

در این مرحله ابتدا تمام سرویس هایی که حداقل یکی از خروجی های سرویس های مرحله I-1 ام را به عنوان ورودی دریافت می کنند انتخاب می کنیم. در این جا سرویس دیگری وجود ندارد که امکان اضافه شدن به گراف را داشته باشد و انتخاب نشده باشد. زیرا اگر چنین سرویسی باشد باید یکی از خروجی های مراحل1 تا I-2 را به عنوان ورودی دریافت کند که این سرویس ها در مراحل قبلی به گراف اضافه شده اند. بنابراین تمام سرویس هایی که در این مرحله قابلیت اضافه شدن به گراف را دارند سرویس هایی هستند که حداقل یکی از خروجی های سرویس های سطح I-1 ام را به عنوان ورودی قبول می کنند. پس از انتخاب شدن این سرویس ها، فیلتر دوم با انتخاب سرویس هایی که تمام ورودی های آن ها از خروجی های سرویس های سطح I-1 ام و سطوح قبلی قابل تامین می باشد، انتخاب می شوند و به گراف اضافه می شوند. در نتیجه با این کار تمامی سرویس هایی که قابلیت اضافه شدن به گراف در سطح I ام را دارند به گراف اضافه می شوند.

از سوی دیگر گسترش گراف محدود می باشد و پس از مدتی به پایان می رسد. زیرا هر سرویس در هر یک از زنجیره های منتهی به آن تنها یک بار ظاهر می شود و بنابراین طول یک زنجیره حداکثر برابر با

تعداد کل سرویس ها n خواهد بود. اگر فرض کنیم حداکثر تعداد خروجی های یک سرویس m باشد در بدترین حالت m^n گره در گراف قرار خواهد گرفت. بنابراین گسترش گراف نهایتا به پایان می رسد و سرویس دیگری به گراف اضافه نخواهد شد.

ثابت شد که گراف کلیه جریان های کاری را که از ورودی های کاربر آغاز می شوند دارا می باشد. در نتیجه گراف کلیه جریان های کاری را که به خروجی های کاربر منتهی می شوند شامل می شود. پس گراف کلیه جواب های کاربر را دارا می باشد.

لم۲: به ازای هر یک از گره سرویس های موجود در گراف که خروجی های کاربر را تولید می کند یک و فقط یک DAG وجود دارد که یکی از جواب های مورد نظر کاربر می باشد.

از آن جایی که گراف ایجاد شده یک گراف متصل می باشد، برای هر یک از گره سرویس های موجود در گراف از گره سرویس آغازین که همان سرویس مجازی کاربر است یک DAG وجود دارد که ورودی های آن را تامین می کند. بنابراین برای هر سرویس موجود در گراف می توان یک DAG در نظر گرفت که با شروع از ورودی های کاربر به فراخوانی گره سرویس کنونی منجر می شود. چنانچه این گره سرویس یکی از خروجی های کاربر را تولید کند، DAG تامین کننده ورودی همراه با گره سرویس تولید کننده خروجی کاربر تشکیل یک جواب کاربر را می دهند. بنابراین حداقل یک DAG که منجر به خروجی های کاربر شود برای هر یک از گره سرویس هایی که خروجی کاربر را تولید می کنند وجود دارد. از آن جایی که هر یک از ورودی های یک گره سرویس تنها از یک گره سرویس دیگر می آید بنابراین حداکثر یک DAG می تواند وجود داشته باشد که ورودی های یک گره سرویس را تامین می کند. با توجه به مطالب گفته شده برای هر گره سرویس در گراف یک و فقط یک DAG وجود دارد که می تواند ورودی های آن را تامین کند. بنابراین به ازای هر گره سرویسی که خروجی کاربر را تولید می کند یک و فقط یک DAG وجود که ورودی های آن را تامین می کند و این DAG همراه با گره سرویس مذکور یکی از جواب های کاربر می باشند.

اثبات قضیه: با توجه به لم۱ الگوریتم تمامی مسیرهایی که از ورودی های کاربر به خروجی های کاربر منتهی می شوند را استخراج می کند و با توجه به لم ۲ برای هر یک از این مسیرها یک DAG وجود دارد که هر یک از این DAG ها یک جواب کاربر می باشد.

۶- نتیجه گیری

در این مقاله یک الگوریتم مبتنی بر گراف برای ترکیب سرویس ها ارائه گردید. در این الگوریتم پس از دریافت درخواست کاربر ابتدا درخواست به کلاس هایی از آنتالوژی‌های مختلف نگاشت می شود. و سپس با توجه به این کلاس های آنتالوژی و رابطه شباهتی که میان کلاس های آنتالوژی در نظر می گیرد به اضافه کردن سرویس ها به گراف می پردازد. الگوریتم ترکیب از سه بخش اصلی جمع آوری سرویس ها، ساخت گراف و پیدا کردن نتایج تشکیل شده است. در بخش جمع آوری سرویس ها، سرویس هایی که در هر مرحله قابلیت اضافه شدن به گراف سرویس ها را دارند جمع آوری می شوند. در هر مرحله سرویس هایی قابلیت اضافه شدن به گراف را دارند که ورودی های آن ها از ورودی های کاربر و یا خروجی های سرویس های قبلی موجود در گراف قابل فراخوانی باشند. در بخش ساخت گراف با استفاده از سرویس‌های جمع آوری شده به ساخت گراف میان سرویس ها می پردازیم. برای جلوگیری از ایجاد حلقه در گراف میان سرویس‌ها این گراف به صورت DAG ایجاد می شود. برای تولید DAG میان سرویسی که در گراف قرار می گیرد و سرویس واقعی تمایز قائل شدیم و بر همین اساس مفهوم گره سرویس را تعریف کردیم که این کار به ما اجازه می دهد بتوانیم از یک سرویس را چندین بار به DAG اضافه کنیم و در زنجیره های مختلف از آن بهره ببریم. در بخش پیدا کردن نتایج با استفاده از DAG ایجاد شده در مرحله قبل یک از یک الگوریتم باز گشتی برای پیدا کردن سرویس های ترکیبی استفاده می‌کنیم. این الگوریتم به صورت جستجوی اول عمق به جستجو در گراف برای پیدا کردن سرویس‌های ترکیبی که می توانند پاسخ کاربر باشند می پردازد. برای نگهداری مسیر طی شده در میان بازگشت های مختلف الگوریتم از یک پشته استفاده می کنیم. در پایان بحث درستی الگوریتم از این لحاظ که می‌تواند تمامی سرویس‌های ترکیبی مورد پاسخ کاربر را تولید کند با استفاده از روش ثابت حلقه اثبات شده است.

مراجع

1. OWL-S Specification www.daml.org/services/owl-s/1.0/
2. OWL Specification, www.w3.org/2004/OWL/
3. Toch E, Gal E, Dori D: Automatically Grounding Semantically-Enriched Conceptual Models to Concrete Web Services. International conference on conceptual modeling(ER 2005) pp 304-319.
4. Klusch, M., Fries, B., Khalid, M., and Sycara, K. 2005. Owls-mx: Hybrid semantic web service, retrieval. In Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web. AAAI Press.
5. B.Arpinar, A.Maduko “Ontology-Driven Web Services Composition Platform” in IEEE International Conference on E-Commerce Technology, 2004, pp. 146-152.

6. B.Medjahed "Sematic web enabled composition of web services, PHD thesis", Virginia Polytechnic Institute, 2004.
7. Fujii, K. and T. Suda, *Semantics-based dynamic service composition* IEEE Journal on Selected Areas in Communications, 2005. **23**(12): p. 2361 – 2372
8. Hashemian, S.V, Mavaddat,F. A graph based approach to web service composition: In proceeding of the 2005 IEEE/IPSJ international symposium on applications and Internet(SAINT), 2005, 183-189.

کارگاه‌های آموزشی مرکز اطلاعات علمی

آموزش نرم‌افزار Word برای پژوهشگران

روش تحقیق کمی

آموزش مهارت‌های کاربردی در تدوین و چاپ مقالات ISI