実世界プログラミングのための分散人力処理環境

匿名で査読を行うため著者名なし *

概要. 概要は和文 600 文字程度で書く.（WISS2012 より概要は和文のみとなった）概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル 概要サンプル

1 はじめに

計算機は多くの問題を解決してきた。計算機によって可能となった高速演算は、人の活動を助け、今ではなくてはならない存在となっている。実世界インタフェースなど、実世界環境とのインタラクションを考慮した仕組みもあり、演算能力だけでなく、実生活にもコンピュータの恩恵は多く入り込んでいる。

一方で、計算機には向いていない問題もある。画像の中の文字認識などは、未だ完璧に行えるわけではない。計算機は、実世界の問題を解決することもまだ不得意である。計算機が実世界環境とインタラクションする際には、センサーやアクチュエータを用いる。しかし、センサーやアクチュエータにも問題がある。センサーは定期的に値を取得することは得意だが、その場の空気を読むといった、コンテキスト依存なセンシングは得意ではない。アクチュエータも、ある一定の動きを繰り返すことは得意だが、一つ一つのアクチュエータの用途は限定的だ。汎用的に使えるようなものではない。

こういったことは、人のほうが得意なことがよくある。人はその場の空気を読んだセンシングも可能である。アクチュエータとして汎用的に実世界に干渉していくこともできる。命令を柔軟に解釈可能であり、いくつかの工程を組み合わせた複雑な処理も可能である。他人との対話が必要な処理に関しては、人がやったほうが良い。今でも、計算機が情報を取得はするが、実際に行動するのは人間、ということはよくあるサンプルを出す

人間の演算能力の部分だけに着目し、人を計算資源とみなして計算機の計算に組み込むようなことがなされているがまた、そのほとんどがクラウドソーシングなどの世界中に分散して存在する不特定多数の人を対象としたものである。身近な人や環境とのインタラクションを考慮した仕組みにはなっていない。

計算機が得意なこともあれば、人が得意なこともある。ならば、計算機が得意なことは計算機が、人が得意なことは人がやるといったように、どちらかの不得意な部分を、どちらかの得意な部分で補うことができれば、現状の計算機・センサ・アクチュエータのモデルではできない、より面白い・便利なことが実現する。そこで、本論文では、実世界プログラミングのためにプログラム上で人を扱えるようにした環境を提案する。

2 実世界プログラミングのための分散人力処理環境

既存の計算機・センサ・アクチュエータによる実世界プログラミング環境では、コンテキストの判断や汎用性といった点において限界がある。本研究では、この問題を解決するために、プログラムと実世界環境とのインタフェースに人を利用できる仕組みを提案する。本提案の仕組みと既存の仕組みを組み合わせることによって、柔軟な実世界プログラミングが可能となる。

2.1 人力利用のメリット

人をプログラムと実世界環境とのインターフェスとして利用することには、大きなメリットがある。人は柔軟な解釈が可能だ。常日頃から、実世界とインタラクションし続けている。実世界とのインタラ

* 匿名で査読を行うため所属名なし

クションをよく知っている高度なコンテキスト判断や汎用性といった点は、計算機・センサ・アクチュエータの既存のモデルでは実現困難である。経験として知っているインタラクションがある

2.2 身近な人・環境とのインタラクション

本提案は、対象を特定の個人や団体とし、身近な人や環境とのインタラクションを記述するためのものである。人の行動をプログラム上で記述する研究は存在するが、クラウドソーシングなどを前提としている。世界中に分散して存在する不特定多数を相手にしており、身近な人や環境とのインタラクションを記述することには向いていない。

2.3 センサ・アクチュエータとの協調

人とセンサ・アクチュエータをプログラム上で同等のオブジェクトとして扱うことで、センサ・アクチュエータなどとの連携を取りやすいような設計にした。プログラム上では、人とセンサ・アクチュエータは同じようなコードで利用可能となっている。実世界プログラミングには、既存のセンサ・アクチュエータの存在が不可欠である。センサーが取得したデータを元に人に行動するよう命令を送ったり、人が取得したデータを元にアクチュエータが処理を行うといったプログラムが考えられる。この時、人とセンサ・アクチュエータの情報交換は、通常のプログラムを書いているのと同じようにできなくてはいけない。

2.4 実現手法

プログラムから Arduino などを介してセンサやアクチュエータを利用する手法を人間にも適応させることで、実世界プログラミングのための分散人力処理環境を実現させる。

1.Arduino などを扱うのと同じように 2. 通常のプログラムの書き方で

例えば、プログラミング言語 Ruby で Arduino を利用する場合、以下のような構文で利用できる。

ソースコード 1. Ruby で Arduino

```
require "arduino_firmata"

arduino = ArduinoFirmata.connect()
res = arduino.analog_read(0)
arduino.analog_write(111, 20)
```

このプログラムでは、以下のようなやりとりや行われている

1. プログラムから arduino に 0 番 PIN のデータを取れ、という命令を送る
2. arduino は命令の通り、0 番 PIN のデータを取得する
3. 0 番 PIN のデータを返り値としてプログラムに返す

このやりとりを抽象化すると、以下のようになる 1. プログラムから処理命令を送る 2. センサ・アクチュエータに処理を実行する 3. 処理の結果をプログラムに返すこの一連の処理の流れを、人にも応用する。以下のようなプログラムが適応させた結果だ。

ソースコード 2. 人力プログラム例

```
require "babascript"

human = Human.new()
res = human.室温はちょうど良いですか()
human.室温を少し下げる()
```

このプログラムでは、以下にようなやりとりが行われている。

1. プログラムから人間に「研究室には今、何人いますか」という命令を送る
2. human は命令どおり、研究室にいる人の数を取得する
3. 人数を返り値としてプログラムに返す

こうすることで、人をプログラム上でセンサ・アクチュエータと同じように扱うための仕組みが実現する。上記のフローを実現するためには、以下の要素が必要となる。

1. プログラム上で人へのタスクを記述可能
2. タスクを適切な人へと配信
3. タスクを人に伝え、処理結果をプログラムに返す

これら 3 要素を組み合わせることで、分散人力処理環境が成立する。1. を満たすために BabaScript を、3. を満たすために BabaScript Client を実装した。2. を満たすために、分散処理プラットフォームの Linda-Base を利用した。

2.5 BabaScript

BabaScript は、プログラム上で人へのタスクを記述可能にするためのライブラリだ。プログラム上でライブラリを読み込むことで、人力構文が利用可能になる。プログラミング言語 Ruby 上で動作する。Baba クラス内で宣言されていないメソッド全てが人への命令として認識される。例えば、to_s というメソッドは全クラスに定義されているため、Baba クラス内では通常の to_s メソッドが実行される。ほげふが というメソッドは定義されていないため、このほげふがというメソッドは人への命令として認識される。この時、タスクとして送信される内容は

- メソッド名

- 引数

である。メソッド名そのものがタスクの内容を示すものであり、引数で補助的に情報を与える。

2.6 BabaScript Client

BabaScript Client は、プログラムから送られるタスクを人に伝え、タスクを処理した結果を入力させ、その内容をプログラムに返り値として返すためのクライアントアプリケーションだ。プログラミング言語 Java を用いて、Android アプリケーションとして実装した。BabaScript Client は、WebSocket を用いて常に Linda-Base と通信を行い、タスクをリアルタイムで受け取り、ワーカーに伝える。

タスクを受け取ると、Android の通知機能を用いてワーカーにタスクの存在を伝える。プログラムから送られてくるタスクは、アプリケーションの起動画面にリスト表示される。ワーカーは、自分に送られてきたタスクの中から実行するタスクを選び、実行する。

ワーカーがタスクを実行したら、実行結果を入力する。ここで入力された内容がプログラム側に返され、タスクの実行完了となる。

BabaScript 側で記述された引数の内容に応じて、実行結果の入力フォームは変化する。返り値の型指定が Boolean 型の場合、true or false だけを表示するフォームに変化し、Int 型の場合、数字を入力するフォームとなる。また、返り値の例示の内容は、フォーム上部に表示され、ユーザの実行結果入力の手助けとなる。

2.7 Linda-Base

Linda-Base は、分散処理のためのプラットフォームだ。Linda という分散処理のための仕組みを Web 上で使える Web サービスで、本研究ではワーカーへのタスク配信のために利用している。タプルスペースという分散グループの単位が存在し、個別のタプルスペースを BabaScript Client が監視することで、タスクを受け取ることができる。例えば、”MasuiLab” というタプルスペースの場合、増井研究室に所属する人に、”TakumiBaba”というタプルスペースの場合、馬場匠見という人物に、BabaScript から送られるタスクが配信される。

本提案では、人への命令を分散可能にした。特定の集団の中の誰かに命令をするための仕組みを容易したかったからだ。例えば、家族の中の誰でもいいから、「洗濯物を取り込んでおいて」という命令を送るなどする時に必要となる。

2.8 処理の流れ

全体の処理の流れは以下の通りである。

1. 人力構文を組み込んだプログラムを実行する
2. Linda-Base を介して、適切な BabaScript Client にタスクが配信される
3. ワーカーにタスクを伝える
4. ワーカーが処理を実行し、その結果を BabaScript Client に入力する
5. 入力された内容が返り値として、プログラムに返される

3 応用例

本論文で提案した分散人力処理環境を利用した応用例を以下に示す。

3.1 ユビキタスコンピューティング

人と計算機・センサ・アクチュエータを組み合わせて使うことによって、実世界ユビキタスコンピューティングとの融合によって、本提案の環境はより強力なものとなる。

3.2 Web サービスの実世界拡張

Web サービスを実世界に拡張することで、今までにないサービスを提供できる。今までの Web サービスは、コンピュータの画面上が主な活躍の場であった。本研究を適応させることによって、Web サービスは画面を飛び越え、実世界へとフィールドを広げることができる。既存のセンサ・アクチュエータとの共存はもちろん、人力だけでも利用できれば、Web サービスの可能性は大きく広がる。BabaScript Client をライブラリ化し、スマートフォンのアプリケーションに組み込めば、既存のアプリケーションの形を維持したまま実世界へのアプローチが可能となる。

4 議論

4.1 求める返り値と実際の返り値の乖離

人は柔軟に解釈可能である。それ故に、プログラム側が求める返り値と、実際にワーカーが返してくる返り値に乖離が起きる可能性がある。人が柔軟に解釈可能である点は、強いメリットでもある。避けられない問題である。

これに対しては、返り値のフォーマットを指定する (ex. 数字のみ) 返り値の例示を行うといったことを行うことで、ある程度は対処可能であると考えられる。ワーカーが適切な返り値を返しやすいようなタスク記述を行うことも、有効な対処法として考えられる。

4.2 応答の確実性

5 関連研究

ヒューマンコンピュテーションは、人を計算資源として捉え、計算機では難しい処理を人に計算させ

るものだ。reCAPTCHA などがその例として挙げられる。人をリソースとして活用する点や計算機には難しい処理を人に委譲する点などにおいて、本研究と類似している。本研究では、人をプログラムと実世界のインタフェースとして利用することを主眼に置いている。

プログラムや Web から不特定多数の人に仕事を依頼するものとして、Amazon Mechanical Turk(MTurk) 等のクラウドソーシングサービスが存在する。TurKit は、通常のプログラムの中で、MTurk のタスクで使われがちな処理をまとめたライブラリ。なおす。AutoMan は、Jabberwocky は CrowdForge では、

Human as Sensor medusa とか

6 おわりに

本論文では、実世界プログラミングのための分散人力処理環境を提案した。計算機では対処しきれない問題はまだ多く存在している。特に、実世界環境とのインタラクションに関しては、より多くの問題が存在している。既存の計算機と実世界環境とのインタラクションの仕組みに、人力を組み込むことで、この問題を解決し、より良い実世界プログラミングが可能となる。

今後は、応用例の実装とその評価を行う。

参考文献

[1] WISS ホームページ. http://www.wiss.org/.

[2] H. Aoki, B. Schiele, and A. Pentland. Realtime Personal Positioning System for Wearable Computers. In *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, pp. 37–43, 1999.

[3] 暦本 純一. まえがき：WISS2000 について. インタラクティブシステムとソフトウェア VIII, pp. i–ii. 近代科学社, 2000.

未来ビジョン

(本行を含む下記の説明を削除してから，記入すること．)

未来ビジョンについては，必須とせず任意とする．論文本体とは別に,「この研究はどういう未来を切り拓くのか」について，著者の視点からアピールしたい点があれば，このような欄を設けて設けて自由に議論してよい．例えば,「こういう未来社会が到来して欲しいから，我々の研究でこう貢献していきたい」,「主張が大きすぎて本文中では書きにくかったが，この研究は，実はこういう気持ちで研究している」,「現在の実装では，小さいトピックであるかのように誤解を招きやすいが，本当はこういう大きなことを狙って，その第一歩として研究に取り組んでいる」のように，研究の未来，魅力を語る場として利用できる．大きさや形状はこのサンプルを目安とするが，この枠内であればある程度改変してもよいものとする．

てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．てすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすとてすと．