You Shall TacItly Understand the Goals of ThIs Paper

Raghu Ranganathan
razetime@pllab.cs.nthu.edu.tw

Abstract
TacIt programmIng, also known as poIntfree programmIng or poIntless programmIng, Is a popular mode of expressIng algorIthms In Array ProgrammIng Languages, and occasIonally other poIntless languages| PoIntfree programmIng styles have been shown to shorten already short code, and often sImplIfyIng programmIng patterns that no one ever wanted sImplIfIed| Hence they are verItable tool In the hands of an array programmer, through the sheer InflexIbIlIty and exactIng nature of them| Here we explore alternatIve approaches to tacIt programmIng In array languages so as to save tacIt programmIng from extInctIon, makIng the useful assumptIon that It requIres savIng In the fIrst place|

IntroductIon

The IntroductIon of tacIt programmIng Into array programmIng began wIth wIth the J programmIng language[1], an exercIse that Involved the thesaurus as much as It dId programmIng language desIgn| The J programmIng language formerly was lImIted to traIns only, and hence It had to provIde flexIble means to compose arbItrary functIons| The basIc tacIt forms of the J programmIng language are popularly dubbed as the *hook* and the *fork*|

$$hook(f, g, x, y) = f(x, g(x, y))$$
$$fork(f, g, h, x, y) = g(f(x, y), h(x, y))$$

J programmIng has strong assocIatIons wIth masochIsm and sadIsm, but the forks and hooks are not the reason why| J uses JuxtaposItIon to form tacIt functIons| The number of functIons dIctates how a traIn Is read| If there Is an odd number of functIons, the traIn Is read as a fork| OtherwIse, It Is read as a hook| ThIs process contInues from left to rIght untIl all functIons are consumed|

$1(+-*)2 = (1 + 2) - (1 * 2) \quad 1(+-)2 = 1 + (-2)$

The hook and fork constItute the fundamentals of tacIt programmIng In modern array programmIng| In order to allow more complex functIon composItIon and hIgher order functIons, there are more provIsIons In array languages| TraIns can be nested wIthIn each other usIng parentheses, and there are ways to “cap” a traIn to avoId It beIng regognIzed as a fork, or omIt arguments| In the J programmIng language, hIgher order functIons can be created usIng a provIsIon called *modIfIer traIns*, whIch are far beyond the scope of thIs paper| These forms altogether can compose to form any arbItrary expressIons, and are capable of TurIng-completeness|

ExIstIng Work

Stack Languages and NIal

Stack languages are tacIt by default| Many stack languages use quotatIons to represent functIons| The specIal case of functIons beIng values that can be manIpulated tacItly allows for very flexIble functIon composItIon and solves many problems wIth array programmIng tacIt systems|

NIal notably Introduces the atlas, tacIt functIon syntax that also allows creatIon of arrays as a part of the system Itself| NIal’s atlases act lIke a “cleave” operatIon upon Its argument(s)|

$[\texttt{abs}, \sin](-\pi) = [\pi, 0]$

SInce these systems provIde very useful, productIve methods of IncorporatIng tacIt programmIng Into array languages, we wIll be convenIently IgnorIng them|

Haskell

Haskell contaIns a poInt free programmIng system wIth poInts| Completely InvalId|

```
join . ((-) .) . (+)
```

Figure 1: poInts? In my poIntfree system?

Not to mentIon that It Is somehow less obvIous than general array language tacIt programmIng|

Jelly

Jelly[2] Is a code golfIng language that contaIns a complex regular expressIon-based tacIt system| The very exIstence of Jelly Is too humourous and too hIgh qualIty for our purposes, and we cannot rIsk further passages beIng remotely funny|

PotentIal Improvements

The maIn problems wIth array tacIt programmIng are dIfferences In readIng from ordInary code, and the number of use cases where they are not applIcable| It may be the case that all array language desIgns of tacIt programmIng are perfect and wIthout flaw| Hence, we Introduce new approaches to Instead understand programmer Intent better|

BogoTraIns

The natural method for recognIzIng the Intent of the user Is to randomIze the way functIons are Interpreted| ThIs can be done In several ways, one of whIch Is Just a plaIn old shuffle and Interpret method| ThIs method shows a $\frac{1}{n!}$ chance of succedIng In InterpretIng the programmer's IntentIon In the average case, whIch Is a margInally hIgher rate of satIsfactIon than what Is currently noted In practIcal use of traIns|

LLMTraIns

ArtIfIcIal IntellIgence tools lIke GItHub CopIlot are famously very Inept at generatIng APL code, but Large Language Models do occasIonally present reasonable soundIng explanatIons when gIven array language code| ThIs can be used to explaIn the programmer's code, but It Is often the case that the programmer has no Idea what they are doIng|

There Is also the case of askIng LLMs to correct APL code, but they seem to be hostIle towards any forms of coherent logIc|

Orchard TraIns

FIg| Tree[3]

A novel approach to tacIt programmIng Is one whIch sImply always comes up wIth an expert's solutIon| We tackle the thIs method the classIcal way: nerd snIpIng|

The APL Orchard (https://apl|chat) Is a chatroom on the Stack Overflow websIte whIch Is frequented by several members of the APL communIty| By askIng a bot to exhIbIt naIvete about a traIn's contructIon, we can gaIn access to useful solutIons In a few mInutes In the average case|

FInal Notes

It would be a crIme for thIs paper to have a poInt| Hence, great efforts have been made to remove all thIngs even mIldly resemblIng poInts from the text of the paper| The understandIng of these specIal, completely purposeful embellIshments Is left as an exercIse for the reader|

Related Work

Great efforts have been made to save tacIt programmIng, notably by the Array Cast crew, who have dedIcated at least 7% of theIr epIsodes (and more parts of of other epIsode) to dIscussIng tacItness In the way thIs research has|

Bibliography

[1] Wikipedia contributors, "APL (programming language)." [Online]. Available: https://en.wikipedia.org/w/index.php?title=APL_(programming_language)&oldid=1213385580

[2] A. A. Hassan, “Effect of royal jelly on sexual efficiency in adult male rats.,” *Iraqi Journal of Veterinary Sciences 23*, 2009.

[3] Maahmaah, “Anjeer Kouhi tree.” [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=21625760