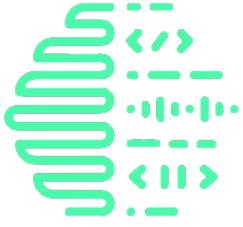
Da Pascal al Pascal

Giuseppe Primiero

UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI FILOSOFIA
"PIERO MARTINETTI"



Un percorso

1. Prendiamo spunto da quattro noti elementi che direttamente o indirettamente si ricollegano ai contributi di Pascal alla matematica, alla computazione e alla filosofia
   - la certezza derivata dalla prova matematica
   - la correttezza della computazione
   - l'incertezza della conoscenza umana

Un percorso

1. Prendiamo spunto da quattro noti elementi che direttamente o indirettamente si ricollegano ai contributi di Pascal alla matematica, alla computazione e alla filosofia
   - la certezza derivata dalla prova matematica
   - la correttezza della computazione
   - l'incertezza della conoscenza umana

2. Ripercorriamo in questa luce il problema della verifica della correttezza dei sistemi computazionali

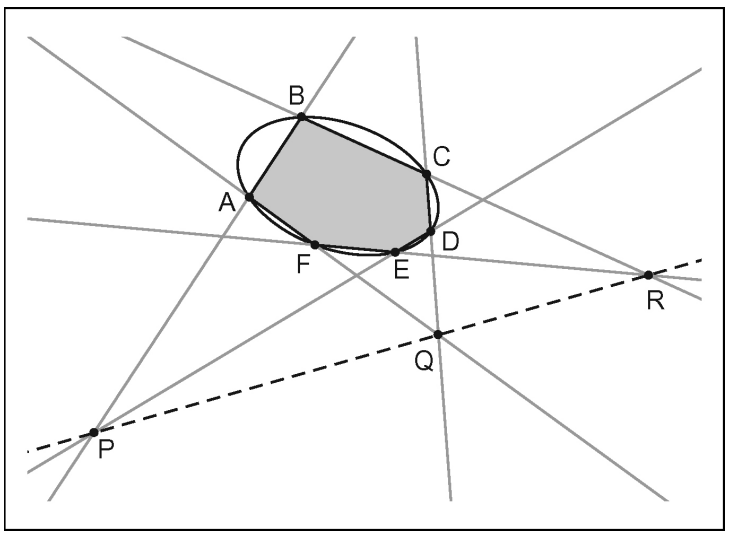
Un percorso

1. Prendiamo spunto da quattro noti elementi che direttamente o indirettamente si ricollegano ai contributi di Pascal alla matematica, alla computazione e alla filosofia
   - la certezza derivata dalla prova matematica
   - la correttezza della computazione
   - l'incertezza della conoscenza umana

2. Ripercorriamo in questa luce il problema della verifica della correttezza dei sistemi computazionali

3. e lo analizziamo nel contesto delle contemporanee tecnologie di intelligenza artificiale.

Mysterium Hexagrammaticum



Teorema

Se un esagono piano $ABCDEF$ è inscritto in una conica, allora le tre coppie di lati opposti $AB - DE, BC - EF, CD - FA$ si incontrano in tre punti allineati; viceversa, se un esagono piano gode di quest'ultima proprietà, allora esso è inscritto in una conica.

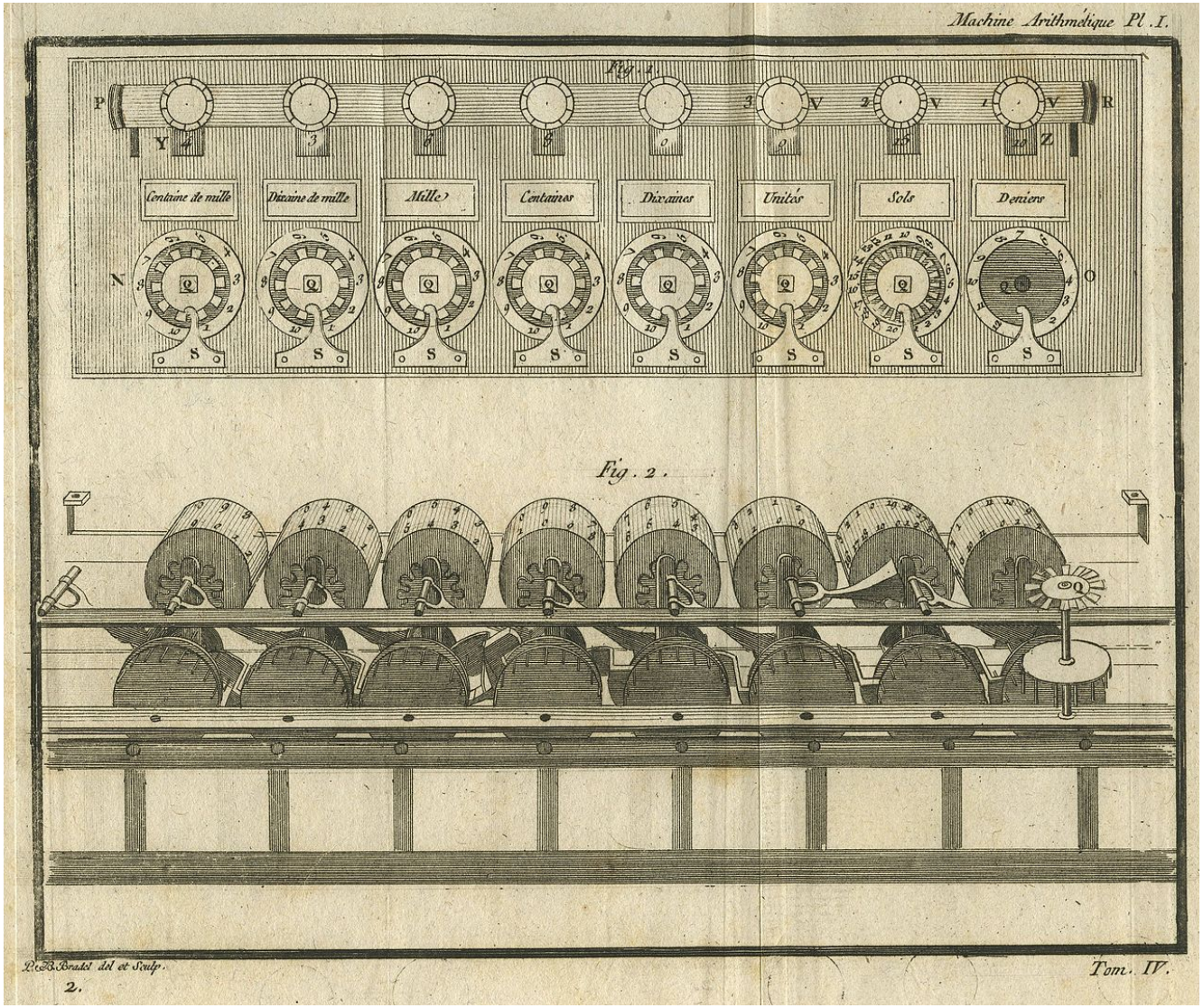
Provare per costruzione

- algoritmi matematici sviluppati dai greci sulla tradizione babilonese ed egizia
- contesto di costruzione nel contesto di problemi pratici
- esempi come il calcolo di fenomeni astronomici tramite l'Antikythera, la predizione delle maree, la balistica, controllo di volo
- esempio classico: bisezione del triangolo

Provare per deduzione

Nello stile algoritmico della dimostrazione matematica, i sottoproblemi che compongono il compito per il quale viene formulato un algoritmo vengono calcolati in modo sequenziale con la possibilità intrinseca di essere riutilizzati in altre combinazioni, una caratteristica che tornerà nella sua interpretazione moderna come subroutine di un programma

Pascalina


Machine Arithmétique Pl. I.
Fig. 1.
Centaine de mille
Dixaine de mille
Mille
Centaines
Dixaines
Unités
Sols
Deniers
Fig. 2.
Tom. IV.
2.

Provare per computazione

Definizione

Un algoritmo è un insieme autonomo, definito passo passo di operazioni che, una volta eseguite, consentono di raggiungere un obiettivo.

Provare per computazione

- processi per la realizzazione dell'intera classe di funzioni computabili
- oltre la matematica pura, gli algoritmi come procedure computabili divengono il fondamento dell'informatica
- una natura ibrida, che combina l'aspetto astratto con l'implementazione

[Knuth, 1997]

"Besides merely being a finite set of rules that gives a sequence of operations for solving a specific type of problem, an algorithm has five important features:

1. *Finiteness. An algorithm must always terminate after a finite number of steps. [. . .]*
2. *Definiteness. Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case. [. . .]*
3. *Input. An algorithm has zero or more inputs. [. . .]*
4. *Output. An algorithm has zero or more outputs. [. . .]*
5. *Effectiveness. An algorithms is also generally expected to be effective, in the sense that its operations must all be sufficiently basic that they can in principle be done exactly and in a finite length of time by someone using pencil and paper."*

Provare le computazioni corrette

Definizione (Problema della Correttezza)

Dato un programma P, è possibile dimostrare se P quando eseguito su un appropriato input i ritorna l'output atteso o, o in altre parole se soddisfa la funzione intesa?

[Floyd, 1967]

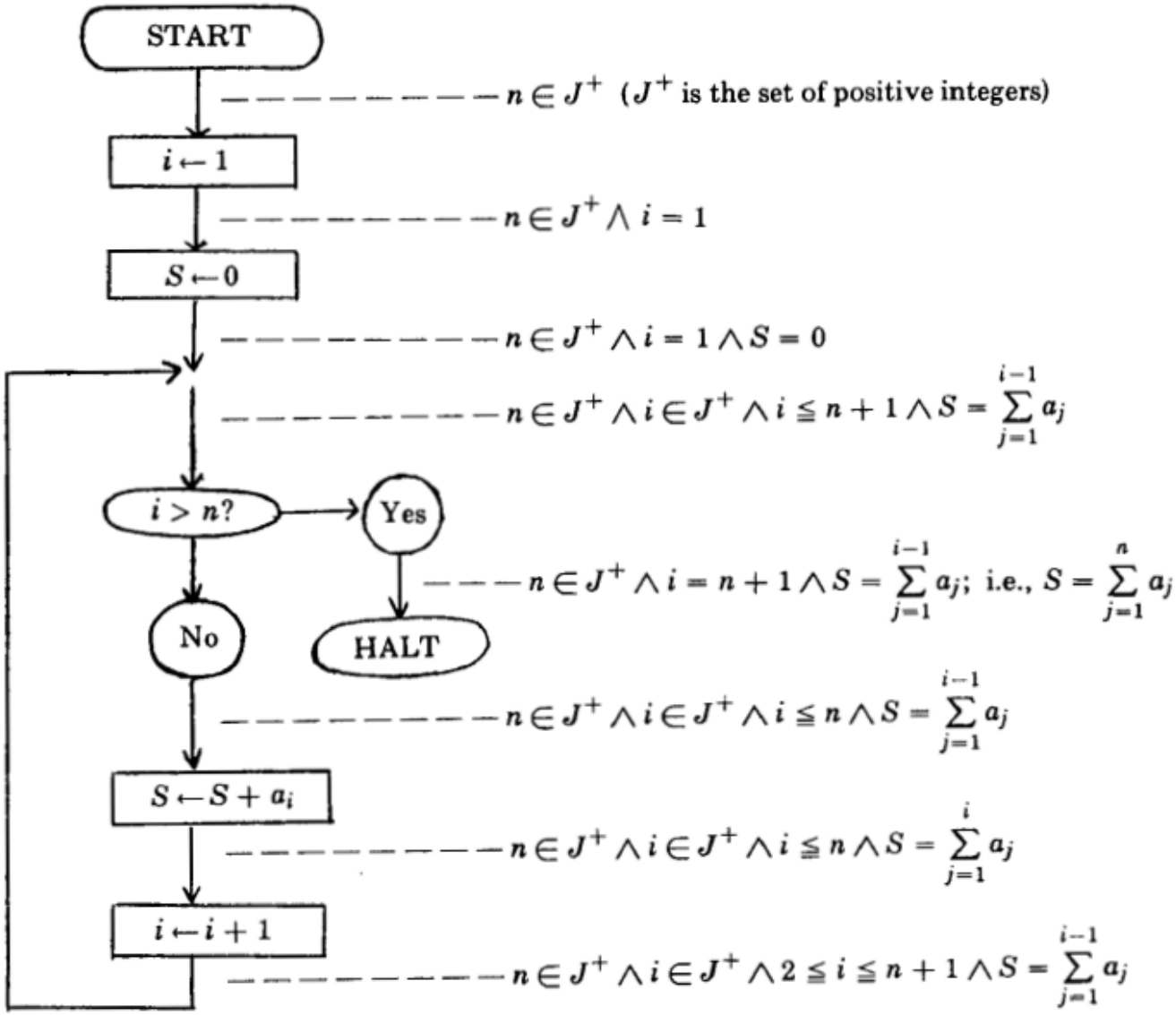


FIGURE 1. Flowchart of program to compute $S = \sum_{j=1}^{n} a_j$ $(n \geqq 0)$

[Floyd, 1967]

- La verifica matematica di un programma è ottenuta tramite la traduzione ad un particolare insieme di assiomi e regole
- Un programma verificato, che ad un qualsiasi passo rende un enunciato vero deve essere seguito ad ogni momento seguente da un passaggio che renda un altro appropriato enunciato vero
- Se il programma termina, il passo per il corrispondente stato di output deve diventare vero.

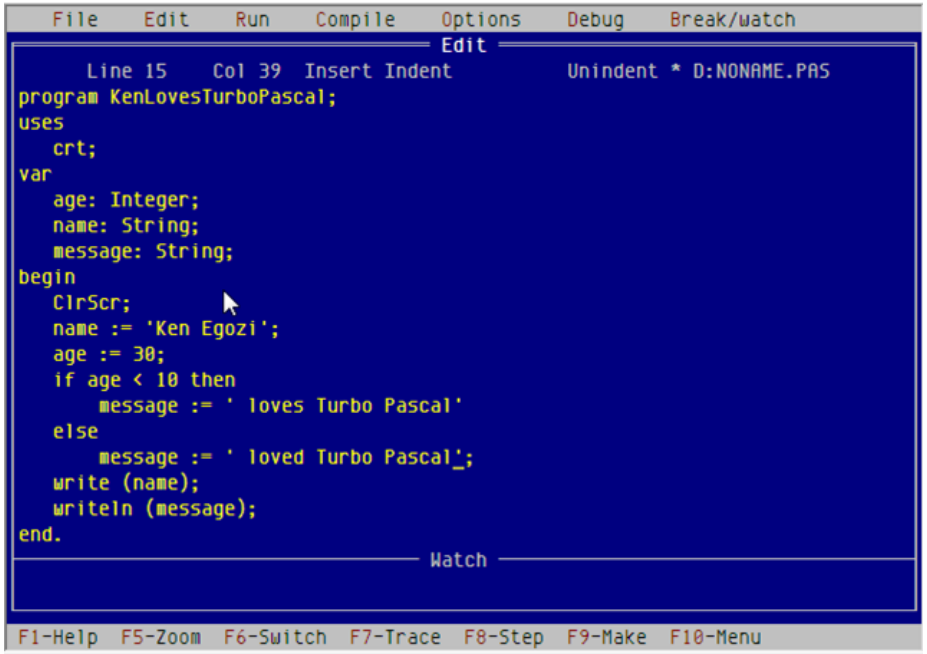
[Floyd, 1967]

Le proprietà dei programmi in un dato linguaggio sono indipendenti dalla loro implementazione fisica e dalla compilazione, e possono essere ridotte allo stabilire standard di rigore per dimostrazione logica sul programma in quel linguaggio.

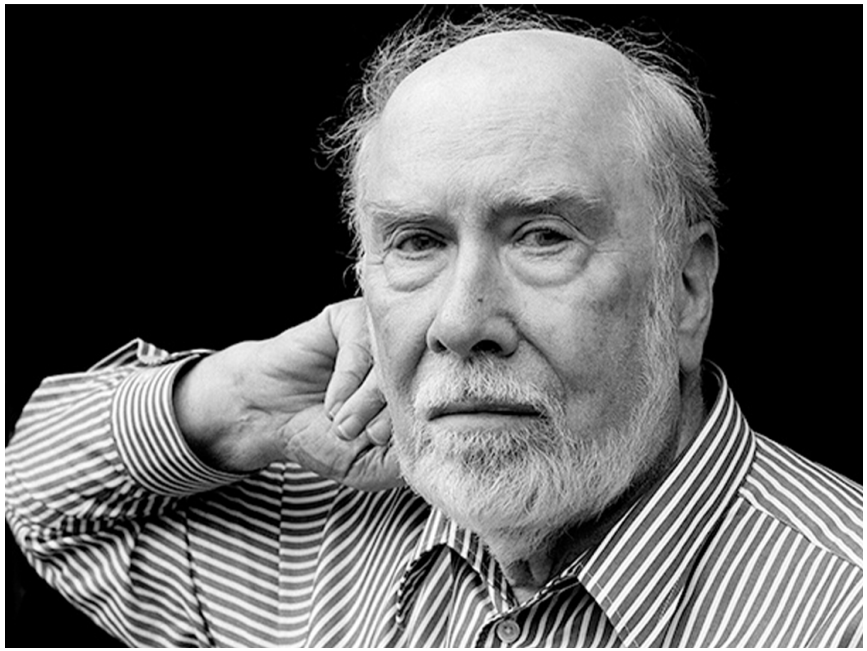
[Hoare, 1969]: program correctness

"Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning. "

Pascal come Linguaggio



Un piccolo esempio



Niklaus Wirth (1934-2024)

Pascal come Linguaggio

- programmazione strutturata (senza GOTO)
- dichiarazione delle variabili
- struttura sequenziale
- etichette, constanti, tipi complessi (record)
- pointers e allocazione dinamica della memoria controllata
- tipizzazione forte per eliminare runtime errors

Dal dominio della Certezza

Sappiamo di non sognare, per quanto ci sia impossibile dimostrarlo con la ragione; questa impossibilità significa che la nostra ragione è debole, non che tutte le nostre conoscenze sono incerte, come essi pretendono. Perché la conoscenza dei primi princìpi, come l'esistenza dello spazio, del tempo, del movimento, dei numeri, è salda come nessuna di quelle che ci danno i ragionamenti, ed è su queste conoscenze del cuore e dell'istinto che la ragione deve appoggiarsi, fondandovi ogni suo ragionamento. [Pensieri, 101]

All'Incertezza epistemica

Ecco la nostra vera condizione. È questo che ci rende incapaci di un sapere certo e di un'assoluta ignoranza. Navighiamo nella vastità, sempre incerti e fluttuanti, spinti da un estremo all'altro. [Pensieri, 185]

Ma controllata

Ogni giocatore ha la certezza del rischio e l'incertezza del guadagno, e tuttavia egli rischia un finito certo per vincere un finito incerto, senza per questo peccare contro la ragione. Non esiste distanza infinita tra la certezza del rischio e l'incertezza del guadagno, ciò è falso. C'è infinità, a dire il vero, tra la certezza della vincita e la certezza della perdita, ma l'incertezza di vincere è proporzionata alla certezza di ciò che si rischia secondo la proporzione dei casi di vincita e di perdita. Da questo deriva che se ci sono uguali possibilità da una parte come dall'altra, la scommessa è alla pari. E allora la certezza di ciò che si rischia è uguale all'incertezza del guadagno, altro che esserne infinitamente distante. [Pensieri, 397]

Che sistemi e che verifica

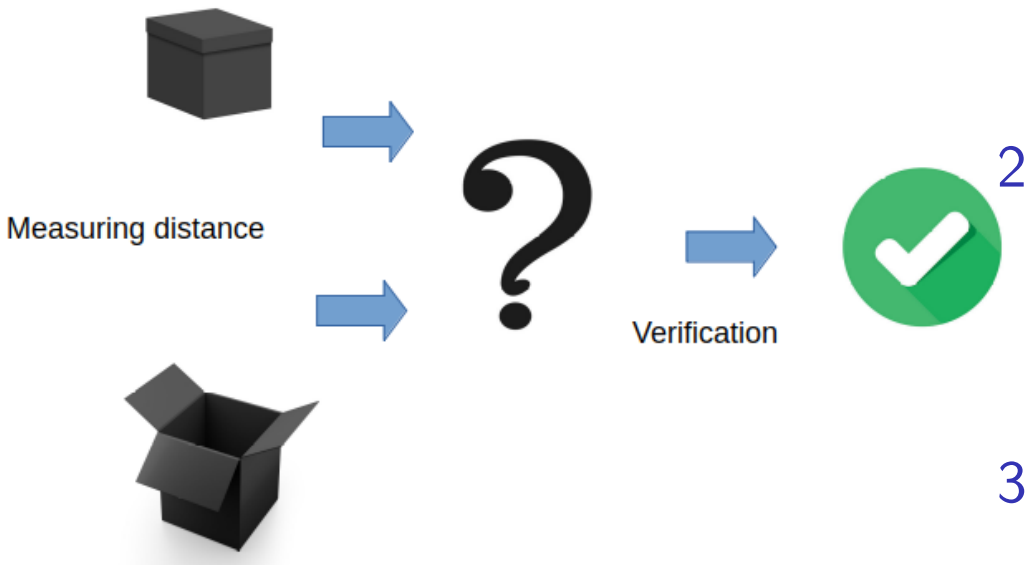
- sistemi la cui specifica non è trasparente: incertezza del comportamento
- instrinsecamente incerti, perchè basati su correlazioni probabilistiche
- richiedono la verifica della loro correttezza, anche in termini di equità e affidabilità

Una domanda logica: verifica

Domanda

Come possiamo verificare che un algoritmo opaco di IA soddisfi certe proprietà, come l'equità, che possano aiutarci ad avere fiducia nel suo uso?

La nostra Strategia [D'Asaro and Primiero, 2021, Primiero and D'Asaro, 2022, D'Asaro et al., 2023]

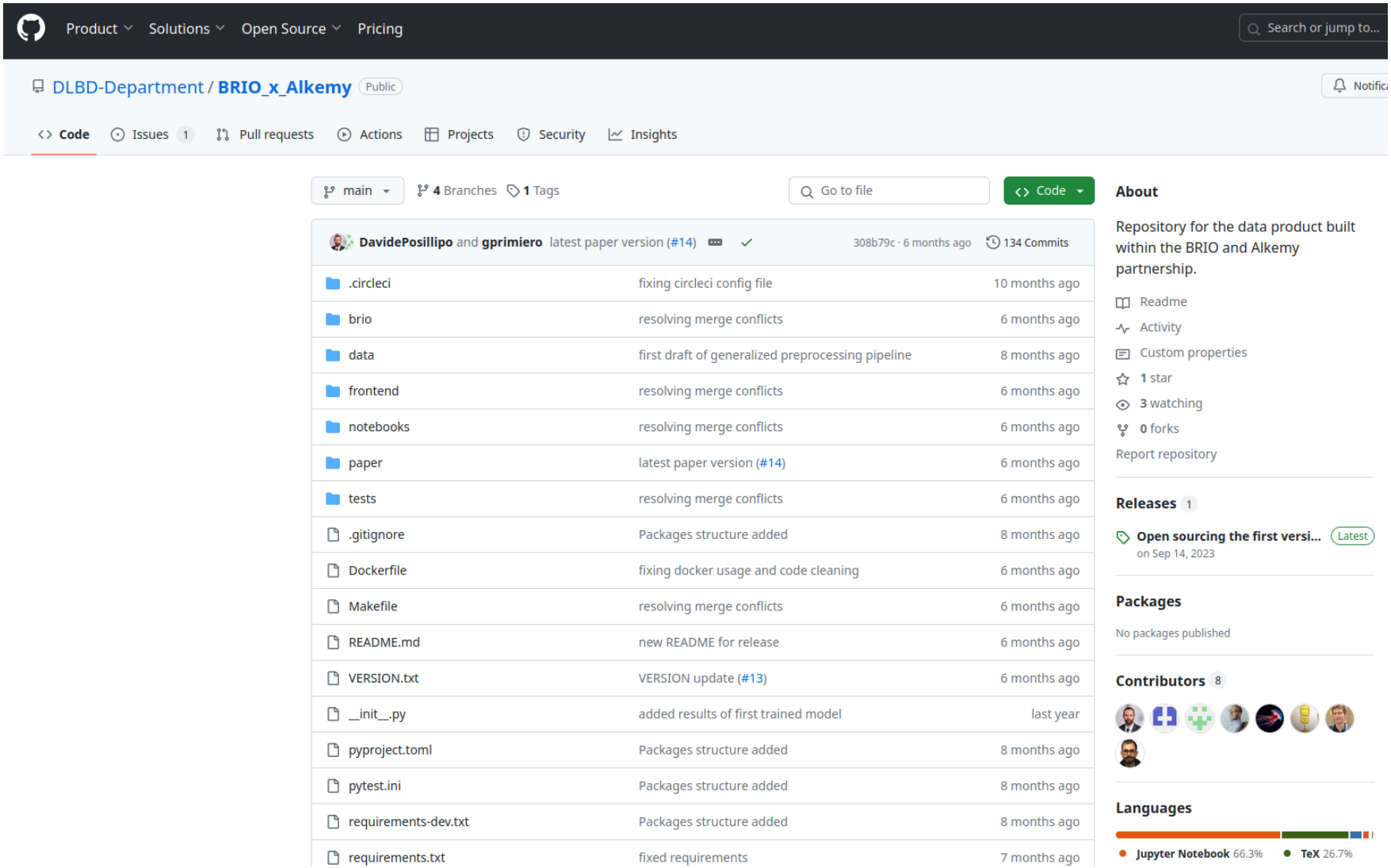


1. Oggetto di analisi è la relazione input/output, senza assunzione di conoscenza strutturale del sistema
2. Si costruisce un modello logico trasparente che esprima un comportamento di riferimento o desiderabile
3. Si misurano appropriate divergenze statistiche tra i due sistemi e se ne analizzano le condizioni di accettabilità

Esempio

Consideriamo un database contenente dati di individui, con la loro età, il sesso e il livello di istruzione. Consideriamo un algoritmo che predice la probabilità di insolvenza del credito. Vogliamo verificare se l'età sia un fattore sensibile in questa previsione. Consideriamo il nostro set di dati, l'output dell'esecuzione dell'algoritmo predittivo e contrassegniamo la caratteristica dell'età come sensibile. Confrontiamo il comportamento dell'algoritmo in relazione all'età rispetto a un comportamento "ottimale" determinato dalla logica (e.g. che ogni gruppo di età abbia lo stesso successo, oppure il comportamento dei diversi gruppi di età rispetto agli altri).

II tool: https://github.com/DLBD-Department/BRIO_x_Alkemy



Analisi

1. confronto tra il comportamento del sistema di intelligenza artificiale e un comportamento desiderabile (espresso come "freq-vs-ref" nell'implementazione), e

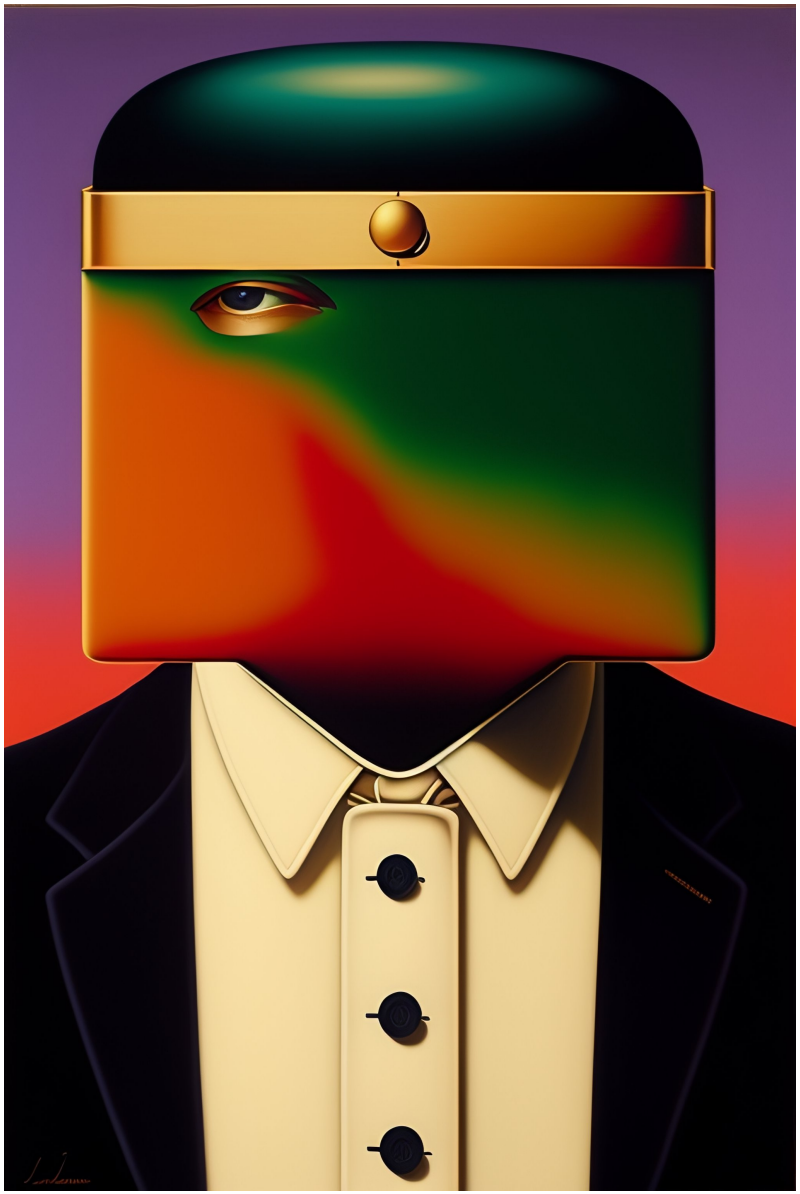
2. confronto tra il comportamento del sistema di intelligenza artificiale rispetto a due classi relativamente alla stessa caratteristica sensibile (espressa come "freq-vs-freq" nell'implementazione);

3. I confronti di cui sopra sono calcolati su distribuzioni di probabilità, ciascuna delle quali esprime il comportamento di un sistema stocastico tramite diverse misure (KL, JS, SV).

Scopo: Verifica Formale del Software

- Controllo del Design: quali sono i comportamenti inequi del sistema?
- Controllo della Safety: il mio output differirà mai significativamente da un comportamento atteso/equo?
- Controllo della Liveness: il mio output resta sempre all'interno di margini di errore ammissibile?

Approssimazione



Dovremmo sempre chiederci per quali caratteristiche un modello computazionale incerto è sia troppo forte che troppo debole, per quali caratteristiche il modello potrebbe essere influenzato positivamente o negativamente. Quali inequità sono ammesse dal sistema.

Modernità di Pascal

1. il rapporto tra la certezza della dimostrazione, la correttezza della computazione e l'incertezza della nostra conoscenza è tornato attuale specie nel contesto di moderne tecnologie

2. Le proprietà che ci serve controllare non sono più solo proprietà matematiche, per il loro impatto umano

3. Dobbiamo sviluppare strumenti che ci aiutino a controllarne i risultati e deciderne gli effetti

4. Le moderne tecnologie di apprendimento data-driven sono per necessità approssimazioni, a noi responsabilità e immaginazione per controllarle.

Grazie dell'attenzione

References I


D'Asaro, F. A., Genco, F., and Primiero, G. (2023).
Checking trustworthiness of probabilistic computations in a typed natural deduction system.


D'Asaro, F. A. and Primiero, G. (2021).
Probabilistic typed natural deduction for trustworthy computations.
In Wang, D., Falcone, R., and Zhang, J., editors, *Proceedings of the 22nd International Workshop on Trust in Agent Societies (TRUST 2021) Co-located with the 20th International Conferences on Autonomous Agents and Multiagent Systems (AAMAS 2021), London, UK, May 3-7, 2021*, volume 3022 of *CEUR Workshop Proceedings*. CEUR-WS.org.


Floyd, R. (1967).
Assigning meanings to programs.
Proceedings of Symposia in Applied Mathematics, 19:19–32.

References II


Hoare, C. (1969).
An axiomatic basis for computer programming.
Communications of the ACM, 12(10):576–580.


Knuth, D. E. (1997).
The art of computer programming, volume 1 (3rd ed.): fundamental algorithms.
Addison Wesley Longman Publishing Co., Inc., USA.


Primiero, G. and D'Asaro, F. A. (2022).
Proof-checking bias in labeling methods.
In Boella, G., D'Asaro, F. A., Dyoub, A., and Primiero, G., editors, *Proceedings of 1st Workshop on Bias, Ethical AI, Explainability and the Role of Logic and Logic Programming (BEWARE 2022) co-located with the 21th International Conference of the Italian Association for Artificial Intelligence*

References III

*(AI*IA 2022), Udine, Italy, December 2, 2022*, volume 3319 of *CEUR Workshop Proceedings*, pages 9–19. CEUR-WS.org.