Compliant-by-construction hybrid medical AI workflows via located task graphs

Vincenzo Ciancia @ CNR-ISTI Emilio Tuosto @ GSSI

ISoLA 2026

Take-away message

Take-away message

- Medical workflows should (by law) be
  - accountable
  - executable
  - "assessed"
  - "localised "

Take-away message

- Medical workflows should (by law) be
  - accountable
  - executable
  - "assessed"
  - "localised "
- LOTS:
  - a lightweight calculus of located task graphs
  - designed for compliance-by-construction

Hush-hush!



Glossing over

- VoxLogicA
- *policies*
- Nearbytes



– Motivation –

Medical protocols

Workflows involving

- datasets...both public and private (subject to institutional boundaries)
- GPUs + PACS[a] + workstations
- nowadays AI + symbolic processing
- human assessment
- controlled circulation of results

[a] Picture Archiving and Communication Systems

Medical protocols

Workflows involving

- datasets...both public and private (subject to institutional boundaries)
- GPUs + PACS[a] + workstations
- nowadays AI + symbolic processing
- human assessment
- controlled circulation of results

[a] Picture Archiving and Communication Systems

Defined in documents

- scripted in natural language
- scattered in local "folders"
- flaky access control

Medical protocols

Workflows involving

- datasets...both public and private (subject to institutional boundaries)
- GPUs + PACS[a] + workstations
- nowadays AI + symbolic processing
- human assessment
- controlled circulation of results

[a] Picture Archiving and Communication Systems

Defined in documents

- scripted in natural language
- scattered in local "folders"
- flaky access control

So what?

This practice is a weak point: usually it produces

- incomplete documentation
- poor support to auditing and repeatability

An example of AI in the loop: VoxLogicA [1]

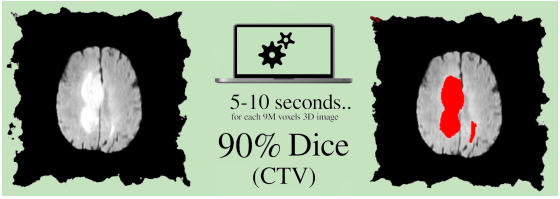
images as spatial models,
analysis as logical specifications in Spacial Logics of Closure Spaces [4],
written in the declarative language ImgQL

background removal
```
load flair = "flair.nii.gz"
let background = touch(intensity(flair) <. 0.1,border)
let brain = complement(background)
```

thresholding
```
let normFlair = percentiles(intensity(flair),brain)
let hyperIntense = filter(5.0,normFlair >. 0.95)
let veryIntense = filter(2.0,normFlair >. 0.86)
```

semantic noise removal
```
let growTum = grow(hyperIntense,veryIntense)
```

texture similarity
```
let tumSim = similarTo(growTum)
let tumStatCC = filter(2.0,tumSim >. 0.6)
let gtv = grow(growTum,tumStatCC)
```

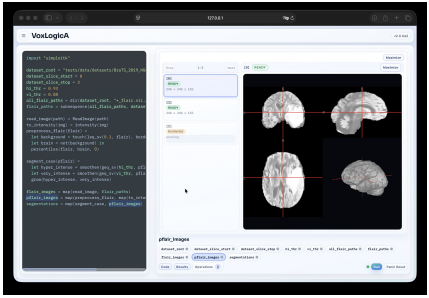


Hybrid AI [2]: neural components segment + symbolic processing + combine + assess

An example of AI in the loop: VoxLogicA (cnt'ed)

A VoxLogicA program interleaves

- data pre-processing
- training neural networks (train via nnU-Net)
- prediction with them (predict)
- post-processing
- harnessing, monitoring, checking
- declarative composition: map, fold, functions



neural and symbolic AI as terms

A regulatory requirement

EU AI Act [6]: medical AI is high-risk

Required practices

- technical documentation
- record keeping
- policies for
  - data governance
  - human oversight
  - accuracy, robustness, cybersecurity

Observation

- These are properties of the workflow
- Compliance is a documentary problem provided that docs "match" the workflow

A natural question

How do we specify workflow with explicit compliance-relevant structure

- derivable from executable specifications
- instead of reconstructing it post-mortem?

– A proposal –

A DSL for compliant-by-construction workflows

A workflow specification language to make explicit & persistent

- locations & dependencies
- ownership boundaries
- assessment steps
- publication events

A DSL for compliant-by-construction workflows

A workflow specification language to make explicit & persistent

- locations & dependencies
- ownership boundaries
- assessment steps
- publication events

"Compliance" should be read as evidence

- *locations, execution, & ownership*
- *dependencies*
- *proper assessment*
- *confidentiality*

are explicit and persistent

A DSL for compliant-by-construction workflows

A workflow specification language to make explicit & persistent

- locations & dependencies
- ownership boundaries
- assessment steps
- publication events

"Compliance" should be read as evidence

- *locations, execution, & ownership*
- *dependencies*
- *proper assessment*
- *confidentiality*

are explicit and persistent

Terms : values, datasets, images, AI models, operations, etc.

Terms and task graphs

LOcated Task graphS: A functional language where invocations are localised operations over computational elements (datasets, files, docs, ...); for instance

$$\text{train}(D)@(\text{store} = \sigma, \text{compute} = \kappa, \text{context} = \omega)$$

trains an AI model on a dataset D at a given location where

Terms and task graphs

LOcated Task graphS: A functional language where invocations are localised operations over computational elements (datasets, files, docs, ...); for instance

$$\text{train}(D)@(\text{store} = \sigma, \text{compute} = \kappa, \text{context} = \omega)$$

trains an AI model on a dataset D at a given location where

- store $= \sigma$ logical address for storage (local dir, remote volume, replicas, ...)

Terms and task graphs

LOcated Task graphS: A functional language where invocations are localised operations over computational elements (datasets, files, docs, ...); for instance

$$\text{train}(D)@(\text{store} = \sigma, \text{compute} = \kappa, \text{context} = \omega)$$

D must be "accessible" from κ!

trains an AI model on a dataset D at a given location where

- store $= \sigma$ logical address for storage (local dir, remote volume, replicas, ...)
- compute $= \kappa$ computational facility for execution (e.g., a GPU in a hospital)

Terms and task graphs

LOcated Task graphS: A functional language where invocations are localised operations over computational elements (datasets, files, docs, ...); for instance

$$\text{train}(D)@(\text{store} = \sigma, \text{compute} = \kappa, \text{context} = \omega)$$

D must be "accessible" from κ!

trains an AI model on a dataset D at a given location where

- store $= \sigma$ logical address for storage (local dir, remote volume, replicas, ...)
- compute $= \kappa$ computational facility for execution (e.g., a GPU in a hospital)
- context $= \omega$ ownership (e.g., institutional, legal, operators, etc.)

Terms and task graphs

LOcated Task graphS: A functional language where invocations are localised operations over computational elements (datasets, files, docs, ...); for instance

$$\text{train}(D)@(\text{store} = \sigma, \text{compute} = \kappa, \text{context} = \omega)$$

D must be "accessible" from κ!

trains an AI model on a dataset D at a given location where

- store $= \sigma$ logical address for storage (local dir, remote volume, replicas, ...)
- compute $= \kappa$ computational facility for execution (e.g., a GPU in a hospital)
- context $= \omega$ ownership (e.g., institutional, legal, operators, etc.)

Why three?

Often $\sigma = \kappa = \omega$, yet they carry different policy information

- a model may be computed, stored, and owned by a hospital
- a prediction's ownership can be computed & stored at a hospital until assessed and forwarded

Terms and task graphs

LOcated Task graphS: A functional language where invocations are localised operations over computational elements (datasets, files, docs, ...); for instance

$$\text{train}(D)@(\text{store} = \sigma, \text{compute} = \kappa, \text{context} = \omega)$$

D must be "accessible" from κ!

trains an AI model on a dataset D at a given location where

- store $= \sigma$ logical address for storage (local dir, remote volume, replicas, ...)
- compute $= \kappa$ computational facility for execution (e.g., a GPU in a hospital)
- context $= \omega$ ownership (e.g., institutional, legal, operators, etc.)

Why three?

Often $\sigma = \kappa = \omega$, yet they carry different policy information

- a model may be computed, stored, and owned by a hospital
- a prediction's ownership can be computed & stored at a hospital until assessed and forwarded

no policy hidden in deployment notes

Assessment as a first-class computation

Ordinary scripted workflows keep human review outside the computation

an expert reads, maybe edits a report, the workflow continues...

LOTS make the review explicit

$$A = \text{assess}(R, P)@\ell_{hospital}$$
$$\text{forward}(R, A, \ell_{recipient})@\ell_{hospital}$$

Prediction/report R, *protocol* P

Forwarding depends on assessment

human responsibility has a place in the graph

Dreaming of compliant-by-construction

Judgements

$$\Gamma \vdash t@\ell : \tau \triangleright \Phi$$

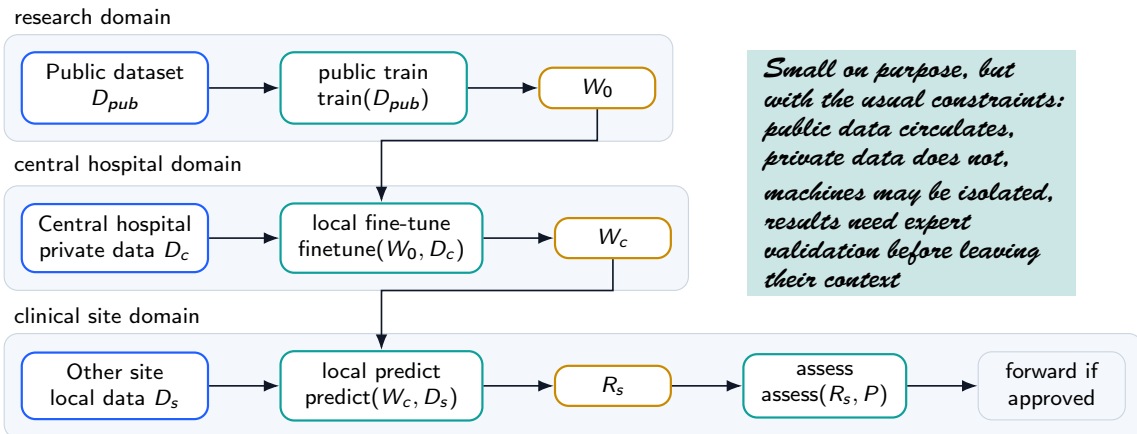
- Γ available datasets, models, credentials, constructed nodes
- τ artefact class produced by t — model, prediction, assessment, ...
- Φ obligations: required assessments, forwarding conditions, key strategies

Status

A notation for policy visibility, not yet a type theory
Next: which contexts are well formed? Which reductions preserve Φ?

– A case study –

Public training, private fine-tuning, local prediction



Small on purpose, but with the usual constraints: public data circulates, private data does not, machines may be isolated, results need expert validation before leaving their context

Located specification: training

Public training

locations as (store, compute, context)
integrity & reproducibility only

$$D_{pub} : \top, \ldots \vdash W_0 = \text{train}(D_{pub})@\ell_{research} : \text{model}$$

Fine-tuning

crosses an institutional boundary

$$D \triangleleft H, \cdots \vdash W_c = \text{finetune}(W_0, D, C)@\ell_H : \text{model}$$

The planner can

- select a hospital-controlled executor
- derive result keys from the hospital dataset policy
- record provenance in the hub log

location and context of data are part of the program

Located specification: prediction, assessment, forwarding

$$D' \triangleleft H', \cdots \vdash R_s = \mathsf{predict}(W_c, D')@\ell' : \mathsf{prediction}$$

forwarding moves storage & context, not computation

Located specification: prediction, assessment, forwarding

$$D' \triangleleft H', \cdots \vdash R_s = \text{predict}(W_c, D')@\ell' : \text{prediction}$$

$$R_s : \text{prediction}, \ldots \vdash A = \text{assess}(R_s) : \text{assessment}$$

forwarding moves storage & context, not computation

Located specification: prediction, assessment, forwarding

$$D' \triangleleft H', \cdots \vdash R_s = \mathsf{predict}(W_c, D')@\ell' : \mathsf{prediction}$$

$$R_s : \mathsf{prediction}, \ldots \vdash A = \mathsf{assess}(R_s) : \mathsf{assessment}$$

$$\begin{array}{ll} \mathsf{case}\ A\ \mathsf{of} & \mathsf{approved}(\alpha) \Rightarrow \mathsf{forward}(R_s, \alpha, H')@\ell' \\ & \mathsf{rejected}(\rho) \Rightarrow \mathsf{stop}(\rho) \end{array}$$

$$A : \mathsf{approved} \vdash \mathsf{forward}(R_s, \alpha, H')@\ell' : \mathsf{forwarding}$$

forwarding moves storage & context, not computation

– Epilogue –

Related work

Located computation

- Klaim [5]: network-aware programming
- AbC [10]: from locations to attributes
- data-oriented behavioural types [3]: consumption, persistence, availability

Logics

- MoMo [12], MoSL [11]: mobility, resources, security
- spatial model checking: topochecker, `VoxLogicA`, PolyLogicA

New: locality + assessment + replayable evidence, in one executable object

Where we are, where we go

Already there

✓ `VoxLogicA-2`: programs → Merkle DAGs, recipes & values in a persistent store, train/predict as primitives
✓ `Nearbytes`: hubs, signed logs, encrypted blocks, replay
✓ `LOTS`: located terms & first-class assessment

Next

- type system, `Nearbytes` implementation `LOTS`
- target properties
  - replay rebuilds the same task graph
  - records refer to existing nodes
  - forwards depend on required assessments
  - peers keep the needed blocks without keys
- validation in research & clinical workflows

Beyond medical imaging: regulated AI workflows in general



Thank you!

References I

[1] Gina Belmonte, Vincenzo Ciancia, Diego Latella, and Mieke Massink. VoxLogicA: A spatial model checker for declarative image analysis.
In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 11427 of *Lecture Notes in Computer Science*, pages 281–298, Cham, 2019. Springer.

[2] Gina Belmonte, Vincenzo Ciancia, and Mieke Massink. Symbolic and hybrid AI for brain tissue segmentation using spatial model checking.
Artificial Intelligence in Medicine, 167:103154, 2025.

[3] Roberto Bruni, Andrea Corradini, Fabio Gadducci, Hernán C. Melgratti, Ugo Montanari, and Emilio Tuosto. Data-driven choreographies à la klaim.
In *Models, Languages, and Tools for Concurrent and Distributed Programming*, volume 11665 of *LNCS*, pages 170–190. Springer, 2019. Essays Dedicated to Rocco De Nicola on the Occasion of His 65th Birthday.

[4] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Model checking spatial logics for closure spaces.
Logical Methods in Computer Science, 12(4), 2016.

[5] Rocco De Nicola, Gian Luigi Ferrari, and Rosario Pugliese. KLAIM: A kernel language for agents interaction and mobility.
IEEE Trans. Software Eng., 24(5):315–330, 1998.

[6] European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence.
Official Journal of the European Union, 2024. Artificial Intelligence Act.

[7] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems.
SIAM Journal on Computing, 18(1):186–208, 1989.

References II

[8] Fabian Isensee, Paul F. Jaeger, Simon A. A. Kohl, Jens Petersen, and Klaus H. Maier-Hein. nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation.
Nature Methods, 18(2):203–211, 2021.

[9] Nearbytes Contributors. nearbytes-app.
Software repository, 2026. Accessed 2026-05-05.

[10] Rocco De Nicola, Tan Duong, and Michele Loreti. Provably correct implementation of the AbC calculus.
Sci. Comput. Program., 202:102567, 2021.

[11] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti, and Mieke Massink. Model checking mobile stochastic logic.
Theor. Comput. Sci., 382(1):42–70, 2007.

[12] Rocco De Nicola and Michele Loreti. A modal logic for mobile agents.
ACM Trans. Comput. Log., 5(1):79–128, 2004.

[13] Michiel Overeem, Marten Spoor, Slinger Jansen, and Sjaak Brinkkemper. An empirical characterization of event sourced systems and their schema evolution – lessons from industry.
Journal of Systems and Software, 178:110970, 2021.