



MFANS 2024 - Dimensional Analysis Made Easy

April 2024

Changing the World's Energy Future

Max Taylor



DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

MFANS 2024 - Dimensional Analysis Made Easy

Max Taylor

April 2024

**Idaho National Laboratory
Idaho Falls, Idaho 83415**

<http://www.inl.gov>

**Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517**

Dimensional Analysis Made Easy

Max Taylor

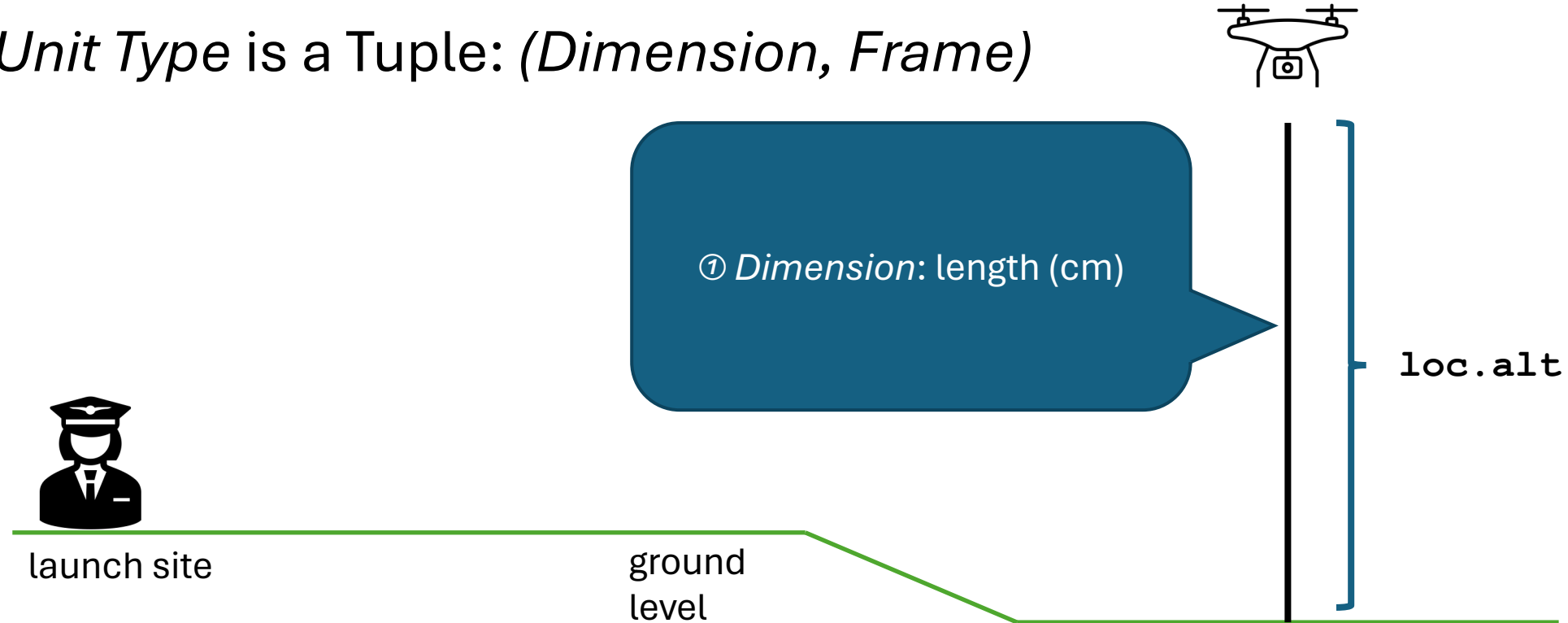
Student @ Ohio State

Formal Methods Intern @ INL

Incoming Formal Methods Scientist @ INL

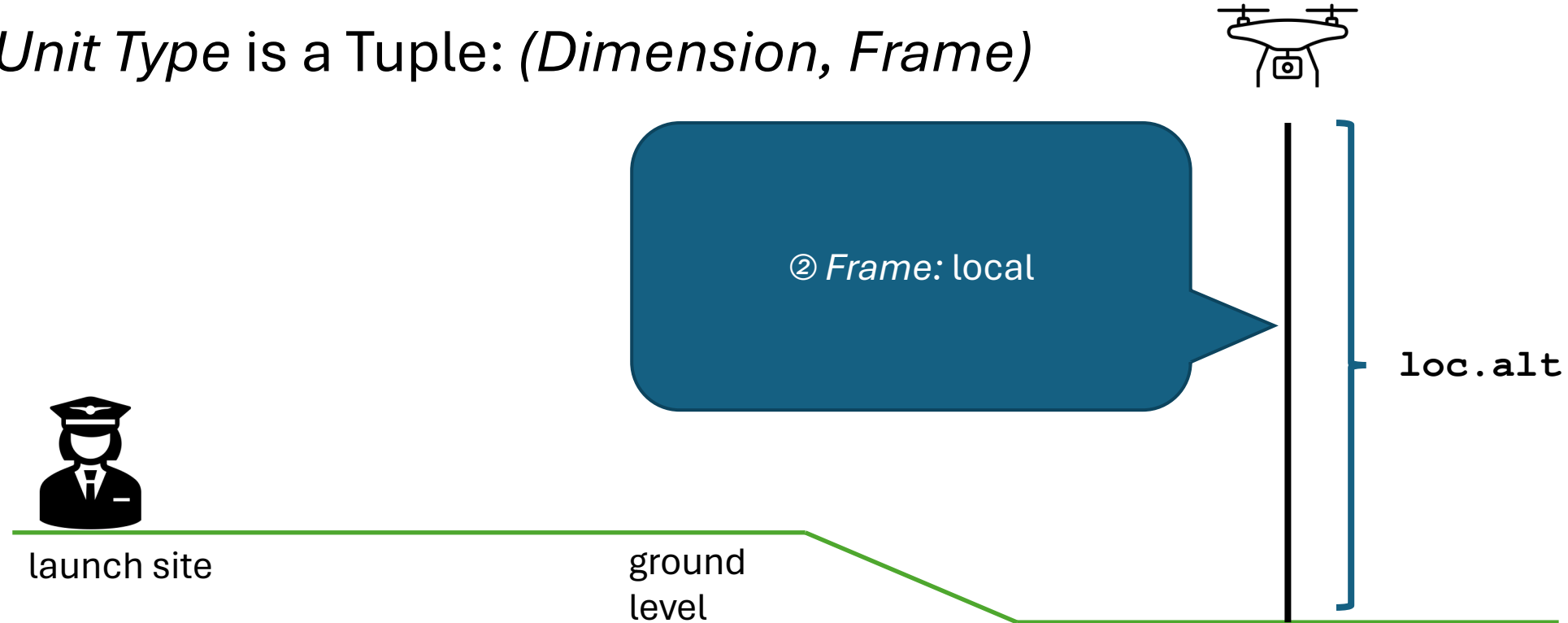
Dimensional analysis

A *Unit Type* is a Tuple: (*Dimension*, *Frame*)



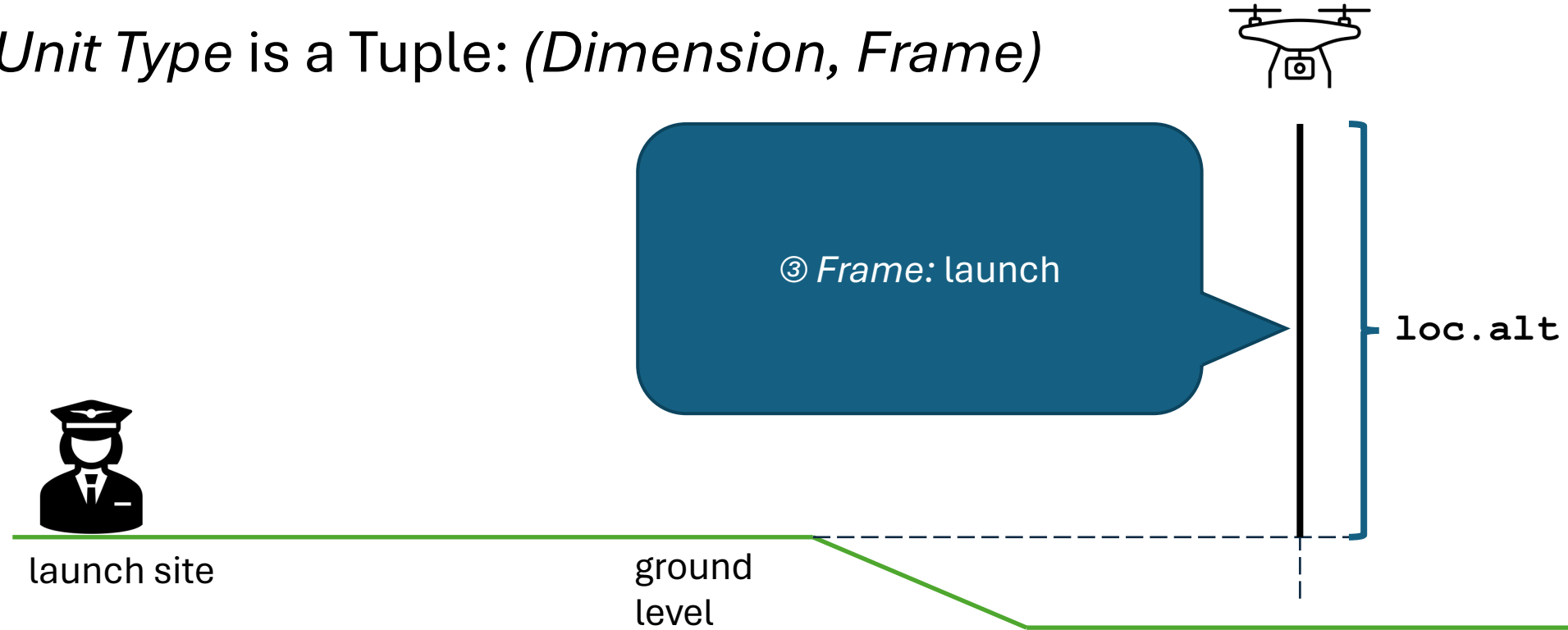
Dimensional analysis

A *Unit Type* is a Tuple: (*Dimension*, *Frame*)



Dimensional analysis

A *Unit Type* is a Tuple: (*Dimension*, *Frame*)



The impact of dimensional errors in systems

```
1. bool far_from_origin(Location &loc) {  
2.     Location ekf_origin;  
3.     if (ahrs.get_origin(ekf_origin) &&  
4.         (ekf_origin.get_distance(loc) >  
5.             EKF_ORIGIN_MAX_DIST_M ||  
6.             ekf_origin.alt - loc.alt >  
7.             EKF_MAX_ALT_M))  
8.         return true;  
9.     return false;  
}
```

Type error:
ekf_origin.alt -
loc.alt has type cm,
expected type m.

The impact of dimensional errors in systems

```
1. bool far_from_origin(Location &loc) {  
2.     Location ekf_origin;  
3.     if (ahrs.get_origin(ekf_origin) &&  
4.         (ekf_origin.get_distance(loc) >  
5.             EKF_ORIGIN_MAX_DIST_M ||  
6.             ekf_origin.alt - loc.alt >  
7.             100.0 * EKF_MAX_ALT_M))  
8.         return true;  
9.     return false;  
}
```

The impact of dimensional errors in systems

```
1. void handle_odometry(mavlink_message &msg) {  
2.     mavlink_odometry m = decode(msg);  
3.     if (m.frame_id != FRAME_LOCAL_FRD) {  
4.         return;  
5.     }  
6.     Vector3f vel{m.vx, m.vy, m.vz};  
7.     handle_vision_speed_est(vel);  
8. }
```

Type error: **vel** has
frame
FRAME_LOCAL_FRD,
expected type
FRAME_LOCAL_NED.

The impact of dimensional errors in systems

```
1. void handle_odometry(mavlink_message &msg) {  
2.     mavlink_odometry m = decode(msg);  
3.     if (m.frame_id != FRAME_LOCAL_FRD) {  
4.         return;  
5.     }  
6.     Vector3f vel{m.vx, m.vy, m.vz};  
7.     handle_vision_speed_est(FrdToNed(vel));  
8. }
```

The impact of dimensional errors in systems

Mystery of Orbiter Crash Solved

By Kathy Sawyer

Washington Post Staff Writer

Friday, October 1, 1999; Page A1

NASA's Mars Climate Orbiter was lost in space last week because engineers failed to make a simple conversion from English units to metric, an embarrassing lapse that sent the \$125 million craft fatally close to the Martian surface, investigators said yesterday.

Why is dimensional analysis hard?

- Systems are large and developers become confused
- Previous solutions UniFi (Hangal et al. 2009), Phriky (Ore et al. 2017), Phys (Kate et al. 2018), PhysFrame (Kate et al. 2021) all:
 - Require manual annotations
 - Struggle with *precise* dimensional analysis
 - Have performance implications that render the tools unsuitable


How to make dimensional analysis easy?

- Use **precise** dimensional representation
- **Infer** the units of variables based on program source code
- **Repair** errors automatically

The resulting approaches are called **SA4U** + **Scalpel** – Tools to analyze C++ codebases

How to represent **dimensions**?

All physical units can be expressed in terms of *base units* defined by the International System of Units (SI)



Unit	Measures
second	time
meter	length
kilogram	mass
ampere	electric current
kelvin	temperature
mole	amount of substance
candela	candlepower

How to represent **dimensions**?

Dimensions are $s \times b_i^{p_i}$ where s is a scalar multiple,

r is a repair constant,

b_i is a SI base unit,

and p_i is a real number



$$cm = r \times \frac{1}{100} \times m^1$$

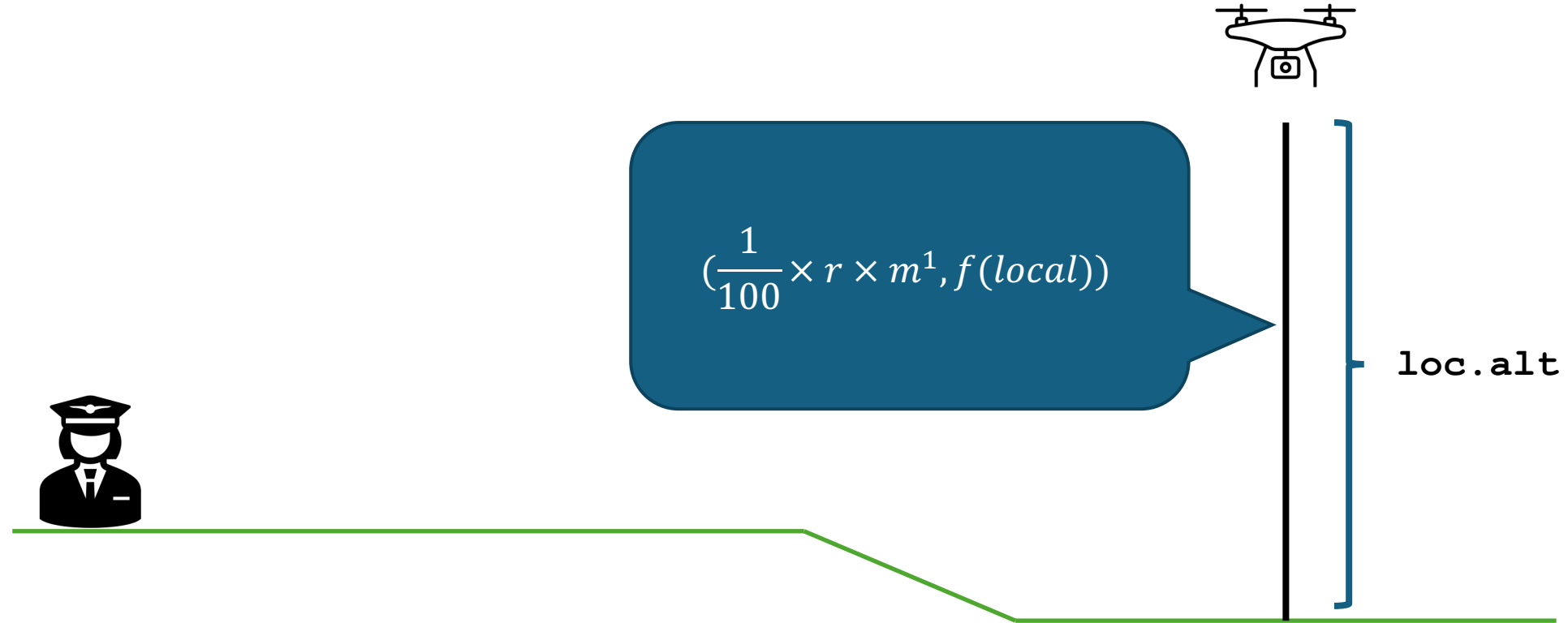


loc.alt

How to represent **frames**?

A *Frame* is a static label that explains **the reference frame where a measurement was obtained**

Type representation



Type algebra

If we have an expression like:

source: $e1 * e2$

model: $e_1: (D_1, F), e_2: (D_2, F)$

We can type it as:

$$e_1 \times e_2: (D_1 \times D_2, F)$$

where

$$(D_1 = s_1 \times r_1 \times b_i^{p_{1,i}}) \times (D_2 = s_2 \times r_2 \times b_i^{p_{2,i}}) = s_1 \times s_2 \times r_1 \times r_2 \times b_i^{p_{1,i} + p_{2,i}}$$

Type constraints

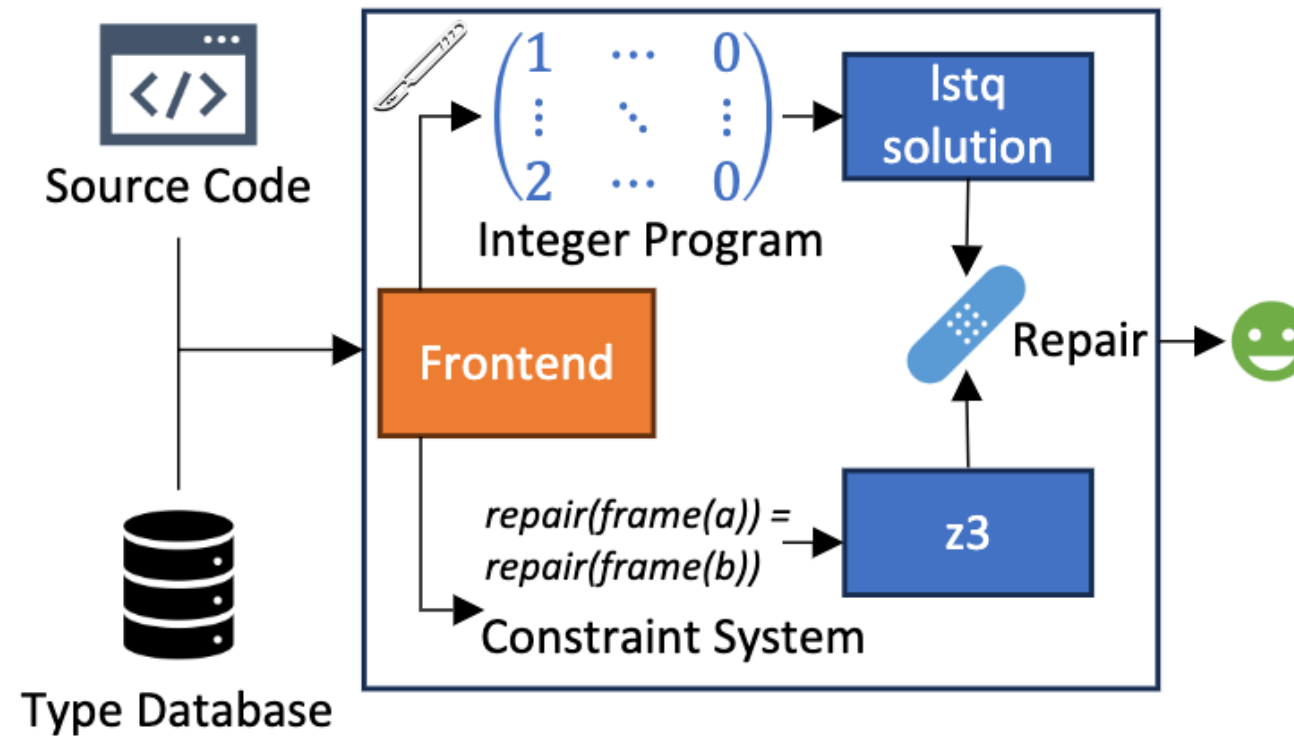
An expression like:

source: $e_1 - e_2$

Imposes the constraint:

model: $type(e_1) = type(e_2)$

How to generate repairs



Conclusion

- SA4U + Scalpel are practical programs to perform **precise** dimensional analysis and **repair** C++ source code
- The dimensional analysis SA4U performs can have an impact beyond traditional software
 - Neuro-symbolic
 - Design verification

Backup slides

How Scalable is SA4U & Scalpel?

UTE Tool	Firmware	LoC	Runtime
SA4U + Scalpel	ArduPilot	848,562	2,215 seconds
SA4U + Scalpel	PX4	197,795	689 seconds
Phys (SoA)	ArduPilot	848,562	4,792 seconds
Phys (SoA)	PX4	197,795	3,951 seconds

Repairing dimensional errors

`ekf_origin.alt - loc.alt > EKF_MAX_ALT_M`

☒ $S_2 = S_3$

`ekf_origin.alt - loc.alt`

$sm = S_0$

$r = \beta_2$

$S_2 = sm + r$

☒ $S_0 = S_1$

`ekf_origin.alt`

$sm = -2$

$r = \beta_0$

$S_0 = sm + r$

`loc.alt`

$sm = -2$

$r = \beta_1$

$S_1 = sm + r$

`EKF_MAX_ALT_M`


$sm = \alpha_1$

$r = \beta_3$

$S_3 = sm + r$

Repairing frame errors

```

    if (m.frame_id != FRAME_LOCAL_FRD)
        $\forall m : \text{mavlink\_odometry} . m.\text{frame\_id} = x \Rightarrow$ 
         $\text{frame}(m.\langle f \rangle) = x$ 
    else
       $\{ \dots \}$ 
       $m.\text{frame\_id} = \text{FRAME\_LOCAL\_FRD}$ 
       $\text{frame}(m.\langle f \rangle) = \text{FRAME\_LOCAL\_FRD}$ 
       $\text{Vector3f vel}\{m.\text{vx}, m.\text{vy}, m.\text{vz}\};$ 
       $\text{frame}(\text{vel}) = \text{frame}(m.\langle f \rangle)$ 
       $\text{handle\_vision\_speed\_est}(\text{vel})$ 
       $\text{frame}(\text{handle\_vision\_speed\_est})$ 
       $= c_0(\text{frame}(\text{vel})) \rightarrow ()$ 
       $(c_0 = \text{id} \wedge c'_0 = 0) \vee (c_0 = \text{FRDToNED} \wedge c'_0 = 1)$ 

```

$$\text{minimize} \left(\sum_i c'_i \right)$$