



GETTING TO TERMS

ETHICS

moral principles that govern a person's behaviour or the conducting of an activity.

MORALITY

principles concerning the distinction between right and wrong or good and bad behavior

BIAS

prejudice in favor of or against one thing, person, or group compared with another, usually in a way considered to be unfair

LAW

system of rules that a particular country or community recognizes as regulating the actions of its members and may enforce by the imposition of penalties.







```
(case (status p)
                                                                                               (Leborri *bornrs* arr onr))
roclaim '(special *depth* *all*
                                                                                              ((setq p (find fn *points*
                                                  (:forgotten nil)
                  *out* *done*))
                                                                                                             :key #'fn-name))
                                                  (:hidden (mapc #'report2 (subs p)))
                                                                                               (report1 (list p) all out))
                                                  (:show
isp:defun report
                                                                                              (t (format out ""%"A is not annotated."
                                                   (cond ((reportable-subs p)
          (&key (fn nil)
                                                                                                         fn))))
                                                          (report3 p)
                (out *standard-output*)
                                                                                            (values))
                                                          (let ((*depth* (1+ *depth*)))
                (all nil))
                                                            (mapc #'report2 (subs p))))
let (p)
                                                                                          (lisp:defun fn-name (p)
                                                         ((reportable p)
(cond
                                                                                            (let ((form (cadr (car (name p)))))
                                                          (report3 p))))))
((not (streamp out))
                                                                                              (and (consp form)
 (with-open-file
                                                                                                    (consp (cdr form))
                                              (lisp:defun reportable (p)
     (s out :direction :output)
                                                                                                    (cadr form))))
                                                (and (eq (status p) :show)
   (report :fn fn :all all :out s)))
                                                     (or *all*
                                                                                            isp:defun report1 (ps *all* *out*)
 ((null *points*)
                                                         (not (= (hit p) *hit*))))
                                                                                             (let ((*depth* 0) (*done* t))
  (format out
                                                                                              (mapc #'report2 ps)
    """No definitions annotated."))
                                              (lisp:defun reportable-subs (p)
                                                                                               (when *done*
 ((not fn)
                                                (and (not (eq (status p) :forgotten))
                                                                                                (format *out*
  (report1 *points* all out))
                                                     (or *all* (not (reportable p)))
                                                                                                   ""%; All points exercised."))))
 ((setq p (find fn *points*
                                                     (some #'(lambda (s)
                                                                                                 (defvar *line-limit* 75)
                 :key #'fn-name))
                                                               (or (reportable s)
  (report1 (list p) all out))
                                                                    (reportable-subs s)))
                                                                                                 (proclaim '(special *depth* *all*
                                                           (subs p))))
 (t (format out ""%"A is not annotated."
                                                                                                                      *out* *done*))
            fn))))
                                              (lisp:defun report3 (p)
(values))
                                                                                                 (lisp:defun report
                                                (setq *done* nil)
                                                                                          ETHES (finite)

(or * tandard-output

(cond)

(cond)
                                                (let* ((*print-pretty* nil)
isp:defun fn-name (p)
(let ((form (cadr (car (name p)))))
                                                       (*print-level* 3)
 (and (consp form)
                                                       (*print-length* nil)
       (consp (cdr form))
                                                       (m (format nil
                                                                   "; "V@T": [-";+"] "{ "S"}"
       (cadr form))))
isp:defun report1 (ps *all* *out*)

(let ((*depth* 0) (*done* t))

(mapc #'report2 ps)
(when *done*

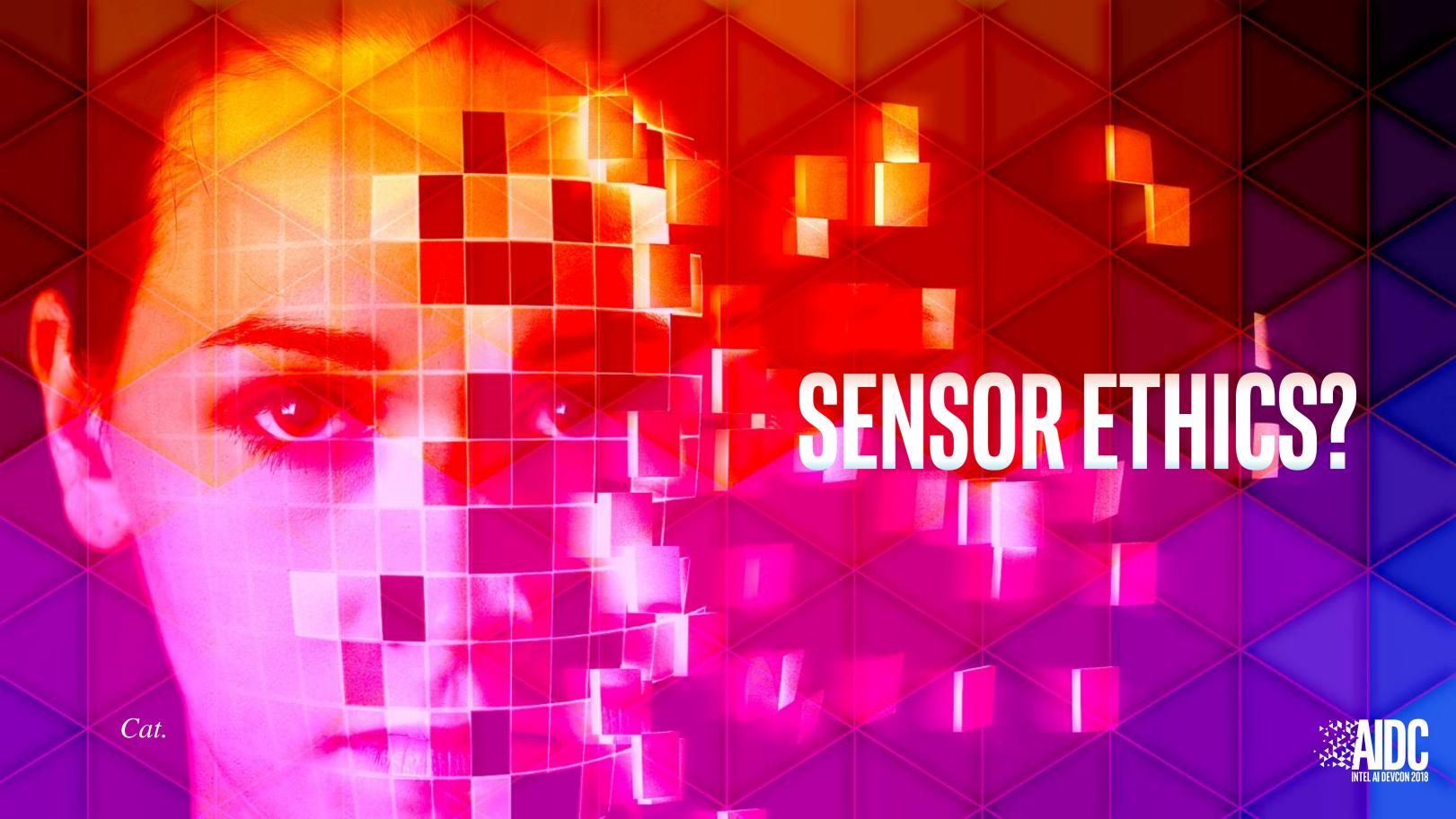
(car (nam p))))

(mapc #'report2 ps)
(when *done*
                                                  (when (> (length m) limit)
                                                                                                      (format out
   (format *out*
                                                    (setq m (subseq m 0 limit)))
     ""%; All points exercised."))))
                                                                                                        ""%No definitions aims
                                                  (format *out* ""%"A <"S>" m (id p))))
                                                                                                     ((not fn)
```

(report1 *points* all out))











```
(case (status p)
                                                                                                            (reporti *points* all out))
roclaim '(special *depth* *all*
                                                                                                           ((setq p (find fn *points*
                                                         (:forgotten nil)
                    *out* *done*))
                                                                                                                             :key #'fn-name))
                                                         (:hidden (mapc #'report2 (subs p)))
                                                                                                             (report1 (list p) all out))
                                                         (:show
isp:defun report
                                                                                                           (t (format out "~%"A is not annotated."
                                                          (cond ((reportable-subs p)
           (&key (fn nil)
                                                                                                                        fn))))
                                                                   (report3 p)
                   (out *standard-output*)
                                                                                                         (values))
                                                                   (let ((*depth* (1+ *depth*)))
                  (all nil))
                                                                     (mapc #'report2 (subs p))))
let (p)
                                                                                                       (lisp:defun fn-name (p)
                                                                 ((reportable p)
(cond
                                                                                                         (let ((form (cadr (car (name p)))))
                                                                   (report3 p))))))
((not (streamp out))
                                                                                                           (and (consp form)
 (with-open-file
                                                                                                                  (consp (cdr form))
                                                    (lisp:defun reportable (p)
      (s out :direction :output)
                                                                                                                  (cadr form))))
                                                       (and (eq (status p) :show)
    (report :fn fn :all all :out s)))
    format out

"""No definitions annotated."))

not fn)
report1 *points* all out))

setq p (find fn *points*

(some #'(lambda (s))

(lisp:defun report1 (ps *all* *out)

#'report2 ps)

*done*

"mat *out*

"some #'(lambda (s)

key #'fn=name))
                                                            (or *all*
                                                                                                         isp:defun report1 (ps *all* *out*)
 ((null *points*)
   (format out
  ((not fn)
   (report1 *points* all out))
                                                                                                                   (All points exercised.")))
 ((setq p (find fn *points*
 ((setq p (find in *points*

(report1 (list p) all out))
(t (format out """ A is not an fin))))

(values))

((setq p (find in *points*

(or (reportable s)

(report1 (list p) all out))

(t (format out """ A is not an *points*

(or (reportable s)

(in '(special *depth* *all*

*out* *done*))

(bkey (fin nil)
                                                                                                                                      *out* *done*))
(values))
                                                                                                                             (&key (fn nil)
                                                       (let* ((*print-pretty* nil)
isp:defun fn-name (p)
                                                                                                                                    out *standard-output
(let ((form (cadr (car (name p)))))
                                                               (*print-level* 3)
                                                                                                                                    (all nil))
 (and (consp form)
                                                                (*print-length* nil)
                                                                                                                (let (p)
        (consp (cdr form))
                                                               (m (format nil
                                                                                                                 (cond
                                                                            "; "V@T": [-";+"] "{ "S"}"
        (cadr form))))
                                                                                                                  ((not (streamp out))
                                                                            *depth*
                                                                                                                   (with-open-file
isp:defun report1 (ps *all* *out*)
                                                                            (= (hit p) *hit*)
(let ((*depth* 0) (*done* t))
                                                                                                                        (s out :direction :output)
                                                                            (car (name p))))
                                                                                                                      (report :fn fn :all all :out s))
 (mapc #'report2 ps)
                                                               (limit (- *line-limit* 8)))
                                                                                                                   ((null *points*)
 (when *done*
                                                         (when (> (length m) limit)
                                                                                                                    (format out
    (format *out*
                                                           (setq m (subseq m 0 limit)))
      ""%; All points exercised."))))
                                                                                                                       ""%No definitions aims
                                                         (format *out* ""%"A <"S>" m (id p))))
                                                                                                                   ((not fn)
                                                                                                                    (report1 *points* all out))
```



