Exemple: condlet

Nous souhaitons pouvoir faire quelque chose comme ceci :

```
* (condlet (((= 1 2) (x (princ 'a)) (y (princ 'b)))
            ((= 1 1) (y (princ 'c)) (x (princ 'd)))
            (t (x (princ 'e)) (z (princ 'f))))
    (list x y z))

CD
(D C NIL)
*
```

Exemple: condlet (suite)

Voici une solution possible :

```
(defmacro condlet (clauses &body body)
  (let ((bodfn (gensym))
        (vars (mapcar #'(lambda (v) (cons v (gensym)))
                      (remove-duplicates
                       (mapcar #'car
                               (mappend #'cdr clauses))))))
    `(labels ((,bodfn ,(mapcar #'car vars)
               ,@body))
       (cond ,@(mapcar #'(lambda (cl)
                           (condlet-clause vars cl bodfn))
                       clauses)))))

(defun condlet-clause (vars cl bodfn)
  `(,(car cl) (let ,(mapcar #'cdr vars)
```

```
                (let ,(condlet-binds vars cl)
                  (,bodfn ,@(mapcar #'cdr vars))))))

(defun condlet-binds (vars cl)
  (mapcar #'(lambda (bindform)
              (if (consp bindform)
                  (cons (cdr (assoc (car bindform) vars))
                        (cdr bindform))))
          (cdr cl)))
```

Exemple: with-db

L'idiome suivant est fréquent :

```
(let ((temp *db*))
  (setq *db* db)
  (lock *db*)
  (prog1 (eval-query q)
         (release *db*)
         (setq *db* temp)))
```

`On souhaite pouvoir écrire~:`

```
(with-db db
  (eval-query q))
```

Exemple: with-db (suite)

```
(defmacro with-db (db &body body)
  (let ((temp (gensym)))
    `(let ((,temp *db*))
       (unwind-protect
         (progn
           (setq *db* ,db)
           (lock *db*)
           ,@body)
         (progn
           (release *db*)
           (setq *db* ,temp))))))
```

Exemple: with-db (suite)

```
(defmacro with-db (db &body body)
  (let ((gbod (gensym)))
    `(let ((,gbod #'(lambda () ,@body)))
       (declare (dynamic-extent ,gbod))
       (with-db-fn *db* ,db ,gbod))))

(defun with-db-fn (old-db new-db body)
  (unwind-protect
       (progn
         (setq *db* new-db)
         (lock *db*)
         (funcall body))
    (progn
      (release *db*)
      (setq *db* old-db))))
```

Exemple: macros anaphoriques

```
(let ((it (complicated-calculation)))
  (if it
      (do-something-with it)
      (do-something-else-with it)))

(aif (complicated-calculation)
     (do-something-with it)
     (do-something-else-with it))
```

Exemple: macros anaphoriques (suite)

```
(defmacro aif (test-form then-form &optional else-form)
  `(let ((it ,test-form))
     (if it ,then-form ,else-form)))

(defmacro awhen (test-form &body body)
  `(aif ,test-form
    (progn ,@body)))

(defmacro awhile (expr &body body)
  `(do ((it ,expr ,expr))
       ((not it))
    ,@body))
```

Exemple: macros anaphoriques (suite)

```
(defmacro aand (&rest args)
  (cond ((null args) t)
        ((null (cdr args)) (car args))
        (t `(aif ,(car args) (aand ,@(cdr args))))))

(defmacro acond (&rest clauses)
  (if (null clauses)
      nil
      (let ((cl1 (car clauses))
            (sym (gensym)))
        `(let ((,sym ,(car cl1)))
           (if ,sym
               (acond ,@(cdr clauses)))))))
               (let ((it ,sym)) ,@(cdr cl1))
```